Deep Hierarchical Reinforcement Learning based Solution for Heterogeneous Swarm Optimization

Siddhant Gangapurwala[1]

***Abstract*—Considering the fact that the problem of heterogeneous swarm optimization, in which robotic systems of different forms perform collaborative tasks to achieve a certain goal, encompasses several research areas of robotics requiring development of solutions for perception, locomotion, navigation and manipulation, this research project combines classical control-theory-based methods for low-level robotic control, such as torque tracking, with deep learning techniques for high-level robotic tasks, such as navigation. This report details upon the work performed for development of a heterogeneous swarm optimization solution using techniques, such as proximal policy optimization (PPO) and trust region policy optimization (TRPO), and further explores the domain of control where both low and high level control tasks are solved using a unified approach of control theory and deep learning. Moreover, this report details the motivation and approach for developing a modular robotic control framework (MRCF) for deploying a heterogeneous swarm system consisting of, but not limited to, quadrupedal robotic systems, autonomous ground vehicles, and autonomous aerial vehicles.**

I. Introduction

As described in [1], the problem of heterogeneous swarm optimization is built on the premise that, in a swarm of homogeneous robots, a single robotic platform cannot cater to all of the aspects of the task at hand, because at the individual level, it is governed by design rules that limit the scope of its capabilities.

The problem of swarm optimization takes into consideration the various sub-domains of robotics research including perception and control, thereby making it a comparatively complex problem for optimization using classical control theory. Moreover, a heterogeneous system further increases the complexity of the problem due to the fact that every different form of robotic platform that exists in the system is controlled by different strategy suited for that particular platform. However, the system of robotic swarm necessitates a generalized approach towards control optimization, since each of the robots in the system contributes towards the realization of a certain goal.

Much of the research on robotic swarm has focused on formation control as in [2] [3] [4] and [5]. Though there has been some work done in swarm distribution based on task allocation as in [6] [7] [8] [9] and [10], the research has either focused on homogeneous swarm of robots or on tasks that cannot be used for practical applications. Moreover, the research work has mostly related to implementing constrained optimization strategies for task allocation based on distribution of traits which further limits the behavior of the robotic systems comprising the swarm. As an alternative, this research project explored the domain of TRPO and PPO techniques [11] to generate a policy for a heterogeneous robotic swarm system to perform a search-and-rescue task.

Previously, researchers, such as authors of [12] and [13], have worked on developing swarm optimization solutions using reinforcement learning (RL) methods but have mainly focused on homogeneous swarm systems. As described by the authors of [10], this report details upon the work on RL approach to solving a cooperative multi-agent task based on locally sensed information of an agent, and extending it to agents of multiple forms.

Despite the tremendous progress in the field of RL, most of the currently developed RL strategies converge to a sub-optimal state as the dimensionality of the problem increases. To develop a solution for a problem such as that of heterogeneous swarm optimization, it is important to consider that the dimensionality of the problem grows exponentially with the number of robotic platforms in the system, thereby, significantly increasing the training period for an RL algorithm, and also, increasing the probability of convergence to a sub-optimal state. Furthermore, if a general policy is implemented for both, executing a cooperative behavior of all the robotic platforms in the system and low-level robotic control, such as for, torque tracking, the dimensionality of the problem further increases, making the problem too complex for training an optimal policy.

This work, therefore, focused on using a hierarchical approach towards high and low level control of the problem. Moreover, a unified method of using classical-control-theory based techniques for low level control and deep learning based techniques for high level control was also tested. The motivation for this research came from previous work done on robotic locomotion using RL methods.

Much of the research in robotic control aims to develop solutions that, depending on the environment of operation, exploit the machine's dynamics in order to achieve agile behavior. This, however, is limited by the use of traditional control techniques such as model predictive control (MPC) [14] and quadratic programming (QP) [15] which are often based on simplified rigid body dynamics and contact models. A model-

[1] Dynamic Robot Systems Group, Oxford Robotics Institute, Department of Engineering Science, University of Oxford, United Kingdom.

[2] Robotic Systems Lab, Institute of Robotics and Intelligent Systems, Department of Mechanical and Process Engineering, ETH Zürich, Switzerland.

This report presents the work performed at the Robotic Systems Lab (RSL), ETH Zürich, as a collaboration between researchers from the Oxford Robotics Institute and the Institute of Robotics and Intelligent Systems for Mini-Project II, as part of the AIMS coursework.

Supervisor: Dr. Ioannis Havoutis[1]. Project performed in collaboration with Jemin Hwangbo[2] and Vassilios Tsounis[2]

based optimization strategy employed over such simplified models often results in a constrained range of solutions that do not fully exploit the versatility of the robotic system, thereby limiting the agility of the robot in question.

Treating the control of robotic systems as an RL problem enables the use of model-free algorithms that attempt to learn a policy that maximizes the expected future (discounted) reward without inferring the effects of an executed action on the environment. Authors of [16] [17] and [18] have successfully implemented these strategies for various robotic 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 reward function in order to learn an optimal policy resulting in a sensible robotic behavior. Often, this reward function needs to be tuned by a human expert. For tasks such as quadrupedal navigation through rough terrain, computing a reward function is also significantly more difficult than for tasks such as posture recovery, which when solved using an RL algorithm results in a near-optimal policy. Hence, a unified approach towards control combining classical-control-theory and deep learning was explored through this project. Furthermore, this project has inspired development of MRCF as a general robotic control framework to be used as a tool by researchers for rapid prototyping of robotic platforms, along with performing comparative analysis of various control algorithms developed by the research community.

This project included testing performance of RL algorithms for a search-and-rescue task by a heterogeneous swarm system comprising of a quadrupedal robotic system, ANYmal [19], and a quadcopter, in which a quadcopter performed the task of exploring the surroundings until it found a goal, after which ANYmal walked towards the goal while avoiding obstacles. The simulations were carried out in *RaiSim* [20] as shown in Figure 1, developed by Jemin Hwangbo[2], Dongho Kang[2] et al. at RSL where this research work was performed, using an RL framework, *RaiLearn*, also developed at RSL.

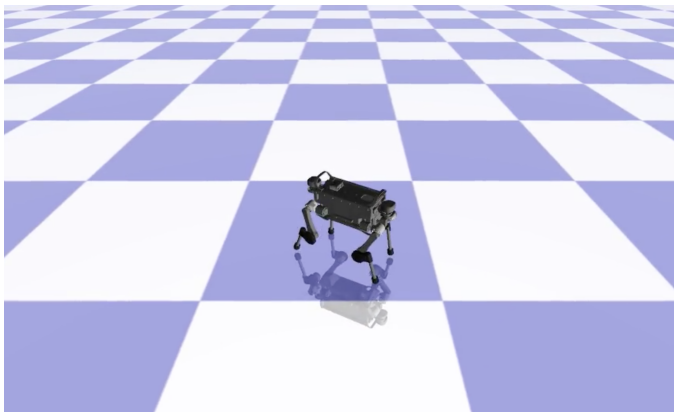

Figure 1. Simulation of the quadrupedal robotic system, ANYmal, on the *RaiSim* simulation platform.

II. Background

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

A. Simulation Platform

The *RaiSim* simulator, used in this project, features efficient recursive algorithms for articulated systems as well as novel contact solvers, as described in [21], thereby, making it a suitable platform for use in data-driven robotics and animation research. The developers of the simulator promise a 50 times speed-up in the training period of an RL algorithm over other simulators such as Gazebo [22]. Also, it has a built-in actuator model. For simulations, the primitive shape objects in the simulator are created using a C++ user API while articulated systems are loaded from URDF. Table I compares some of the widely used simulators, including Bullet [23], ODE [24] and MuJoCo [25], detailed in [26].

Table I

	RaiSim	**Bullet**	**ODE**	**MuJoCo**
Initial Release	Unreleased	2006	2001	2015
Author(s)	J. Hwangbo D. Kang	E. Coumans	R. Smith	E. Todorov
License	Proprietary	Zlib	GPL/BSD	Proprietary
Main Purpose	Robotics	Gaming	Gaming	Robotics
Language	C++	C/C++	C++	C
API	C++	C++/Python	C	C
Contacts	Hard	Hard/Soft	Hard/Soft	Soft
Solver	Bisection	MLCP	LCP	PGS/CG
Coordinates	Minimal	Minimal	Maximal	Minimal

Some of the benchmark results for the following tests -

- **Rolling test:** Friction model test.
- **Bouncing test:** Single-body elastic collision test.
- **666 balls test:** Single-body hard contact test.
- **Elastic 666 balls test:** Single-body energy test.
- **ANYmal PD control test:** Articulated-robot-system speed test for quadrupedal robot.
- **ANYmal momentum test:** Articulated-robot-system momentum test.
- **ANYmal energy test:** Articulated-robot-system energy test.

are as shown in Table II.

Table II

	RaiSim	**Bullet**	**ODE**	**MuJoCo**
Rolling	++	+++	-	+
Bouncing	++++	++	+++	-
666	+++	+	++	+
Elastic 666	++++	++	+++	-
ANYmal PD	+++++	+++	+	++++
ANYmal Momentum	+++	++	+++++	++++ (RK4) ++ (Euler)
ANYmal Energy	++++	+++	++	+++++ (RK4) +++ (Euler)

where more '+' is better, and '-' refers to - cannot be simulated due to inaccurate model or excepted. For its performance, it is expected that *RaiSim* will be a good alternative to the current state-of-the-art engines for contact simulation of robotics.

B. Learning Framework

Similar to *RaiSim*, the learning framework *RaiLearn* was developed at RSL to be integrated with the RAI system. *RaiLearn* is a C++ framework built to develop and benchmark learning algorithms. It has been designed for reinforcement tasks especially for robotic applications. It is basically a collection of classes that are useful for learning and are categorized into the core module and non-core modules. The core module contains classes that are essential for reinforcement learning, such as algorithms, noise, tasks, and memory, and non-core modules contain useful tools for coding, such as graphics and utilities. Implementation of a new environment in *RaiLearn* is similar to that of *OpenAI Gym* [27]. The framework is not yet publicly released.

C. Trust Region Policy Optimization

Effective for optimizing large nonlinear policies, such as neural networks, TRPO [28] has demonstrated robust performance on a wide variety of tasks, such as learning simulated robotic swimming, hopping and walking gaits.

As described in [29], trust region methods define a region around the current iterative within which they trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this region.

Consider a Markov Decision Process (MDP) given by the tuple $(S, A, \{P_{sa}\}, \gamma, R, \rho_0)$ where

- S is a finite set of N states
- $A = \{a_1, ..., a_k\}$ is a set of k actions
- $P_{sa}\left(s^{'}\right)$ is the state transition probability of landing at state $s^{'}$: $P\left(s, a, s^{'}\right)$ upon taking the action a at state s
- $\gamma \in [0, 1)$ is the discount factor
- $R : S \rightarrow \mathbf{R}$ is the reward function
- $\rho_0 : S \rightarrow \mathbf{R}$ is the state distribution of the initial state s_0
- $\rho_\pi : S \rightarrow \mathbf{R}$ is the discounted visitation frequencies,

$$\rho_\pi = Pr\left[s_0 = s\right] + \gamma Pr\left[s_1 = s\right] + \gamma^2 Pr\left[s_2 = s\right] + ...$$

- $\eta(\pi) = \mathbf{E}_{s_0, a_0, ..}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s_{t+1}\right)\right]$ is the expected discounted cumulative reward of policy π
- $Q_\pi\left(s_t, a_t\right) = \mathbf{E}_{s_{t+1}, a_{t+1}, ..}\left[\sum_{l=0}^{\infty} \gamma^l r\left(s_{t+l}\right)\right]$ is the action value function
- $V_\pi\left(s_t\right) = E_{a_t, s_{t+1}, ..}\left[\sum_{l=0}^{\infty} \gamma^l r\left(s_{t+l}\right)\right]$ is the value function
- $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$ is the advantage function

As derived in [30],

$$\eta(\pi) \approx \eta\left(\pi_0\right) + \mathbf{E}_{\rho_{\pi_0}} \mathbf{E}_{a \sim \pi(s)}\left[A_{\pi_0}(s, a)\right]$$

The TRPO problem (Algorithm 1), as detailed in [28], is then given by

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & L_{\theta_0}(\theta) - C\bar{D}_{KL}\left(\pi_0 || \pi\right) \\ \text{subject to} \quad & \bar{D}_{KL}\left(\pi_{\theta_0} || \pi_\theta\right) \leq \delta \end{aligned}$$

Algorithm 1 Approximate policy iteration algorithm guaranteeing non-decreasing expected return η

1: Initialize π_0.
2: **for** $i = 0, 1, 2, ...$ until convergence **do**
3: Compute all advantage values $A_{\pi_i}(s, a)$.
4: Solve the constrained optimization problem

$$\pi_{i+1} = \arg\max_{\pi} \left[L_{\pi_i}(\pi) - \left(\frac{2\epsilon'\gamma}{(1-\gamma)^2}\right) D_{KL}^{max}\left(\pi_i, \pi\right)\right]$$

$$\text{where } \epsilon' = \max_s \max_a \left|A_\pi(s, a)\right|$$

$$\text{and } L_{\pi_i}(\pi) = \eta\left(\pi_i\right) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$$

5: **end for**

D. Proximal Policy Optimization

Unlike in the case of TRPO, PPO gets rid of the computation created by constrained optimization by proposing a clipped surrogate objective function as detailed in [31]. The PPO algorithm that uses fixed length trajectory segments is shown in (Algorithm 2).

Algorithm 2 Proximal Policy Optimization using Actor-Critic Method

1: **for** iteration=1,2,... **do**
2: **for** actor=1,2,...,N **do**
3: Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
4: Compute advantage estimates $\hat{A}_1, ..., \hat{A}_T$
5: **end for**
6: Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$; $\theta_{\text{old}} \leftarrow \theta$
7: **end for**

III. Problem Statement

The motivation for the work detailed in this report was the fact that much of the research on robotic swarm systems focused on homogeneous swarms or on use of constraint based optimization strategies. 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 heterogeneous swarm optimization including task allocation and low-level control such as robotic locomotion, the research work focused on using a hierarchical reinforcement learning strategy for both high and low level control of the robotic platforms.

The work carried out as part of this project can be stated as: *Developing a hierarchical reinforcement learning based strategy for high and low level control of robotic platforms including a quadrupedal robot and an aerial vehicle comprising a heterogeneous swarm system for execution of a search-and-rescue task.* Moreover, experiments implementing a unified approach for high and low level control based on deep learning and classical control theory methods were also performed.

IV. Approach

The overall flow of the project consisted of:

1) Design and development of simulation environment for *RaiSim* simulator
2) Integration of the developed environment with *RaiLearn* framework
3) Testing the performance of the RL strategies for different tasks and reward functions

A. Design and Development of Simulation Environment

The *RaiSim* simulator, being in the early stage of development, required further development to include robotic platforms such as aerial vehicles. The simulator does not yet support propulsion simulation. Therefore, using an existing quadcopter model was not possible. Instead, a box was used to emulate quadcopter dynamics governed by

$$\tau_B = \begin{bmatrix} Lk\left({\omega_1}^2 - {\omega_3}^2\right) \\ Lk\left({\omega_2}^2 - {\omega_4}^2\right) \\ b\left({\omega_1}^2 - {\omega_2}^2 + {\omega_3}^2 - {\omega_4}^2\right) \end{bmatrix},$$

where τ_B refers to the torques in the body frame of the quadcopter, L is the distance from the center of the quadcopter to any of the propellers, and ω_i is the angular velocity of the motor i. The total thrust on the quadcopter in the body frame is given by

$$T_B = \sum_{i=1}^{4} T_i = k \begin{bmatrix} 0 \\ 0 \\ \sum {\omega_i}^2 \end{bmatrix}.$$

The ANYmal quadrupedal robotic system was also used in the environment. The quadcopter task included exploring the environment until it found a goal, after which, ANYmal started walking towards the goal while avoiding obstacles. The quadcopter updated the global map at each iteration. The global map included information such as explored region, location of obstacles, traversable region, and location of the goal. Since, the $RaiSim$ simulator, does not include a camera capture feature, a 50×50 matrix was used to generate a global map representing the $(15 \times 15)\, m^2$ area of operation. -1 represented unexplored region in the global map. 0 referred to a traversable region and 1 was used to represent a region consisting of obstacles. This map was updated at every time step depending on the location of the quadcopter.

The simulation environment is as shown in Figure 2.

B. Integration with Learning Framework

The simulation environment developed for *RaiSim* was then used to integrate with the *RaiLearn* learning framework. The software architecture of the framework shares resemblance with *OpenAI Gym*, except *RaiLearn* only supports C++. Moreover, it uses *TensorFlow* [32] as a backend for initializing computational graphs.

The framework provides algorithms such as PPO, TRPO, deep deterministic policy gradients (DDPG), etc. that were used to train some of the control policies in the swarm system as described in the next sub-section.

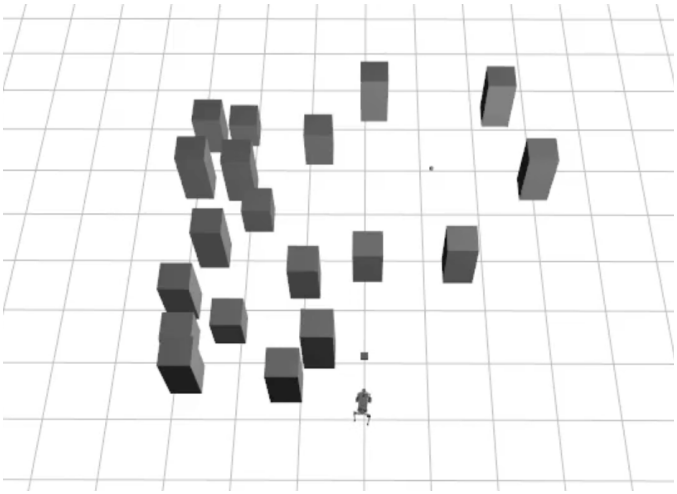

Figure 2. RAI-Sim environment for simulating a heterogeneous swarm system consisting of an ANYmal quadruped and a quadcopter (here represented as a floating box). The quadcopter first explores the map until it discovers the goal (shown as a sphere), after which, ANYmal starts to walk towards the goal while avoiding the obstacles.

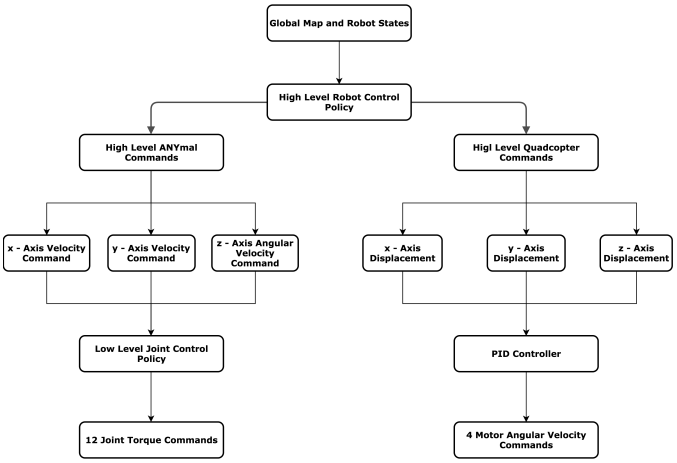

Figure 3. The control strategy used for both high and low level control of the robotic platforms in the heterogeneous swarm system.

C. Training and Testing Control Policies

The hierarchical robot control strategy for the search-and-rescue task is as shown in Figure 3. Both the robotic platforms are given high level commands based on the robot states and the updated global map from a trained policy. The high level commands given to the ANYmal quadruped include $\dot{x}$, $\dot{y}$ and $\dot{\psi}$. These commands are given to a neural network trained using a policy gradient technique which outputs joint torque commands for each of the 12 joints in the quadruped. The high level commands to the quadcopter are given to a PID controller [33] which then outputs angular velocities for each of the motors of the quadcopter.

1) Low-Level Robot Control: The low level ANYmal control was done using a neural network trained using TRPO. Both the value function and policy estimation was done using multi-layer perceptron (MLP) network as represented in Figure 4. A comparatively simple network managed to

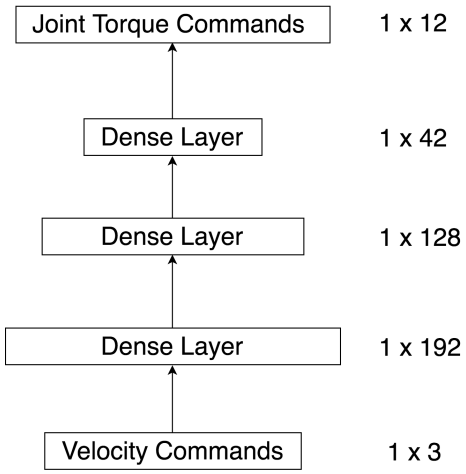

Figure 4. The architecture of the MLP used to train the policy for joint torque control of the ANYmal quadruped.

generate a complex policy for quadruped locomotion using the TRPO method. In fact, the gait obtained using the trained policy was observed to be 25% faster than other hand-coded control strategies. The policy also performed extremely well for stabilization when an external force was applied to the quadruped. The trained policy was thus used for low level control of the ANYmal robot.

The quadcopter controller was fairly simple. PID control technique was used to generate angular velocity commands for each of the motors of the quadcopter. The input to the controller included x, y and z axis displacements. These were obtained from the high-level control policy.

2) High-Level Robot Control: A general policy was trained to perform a cooperative multi-agent search-and-rescue task using the PPO method. The neural network architecture, represented in Figure 5, was used for both policy and value function estimation of the search-and-rescue task. The computation graph was created using *TensorFlow* which was then used along with the environment created for *RaiSim* for training. Simple experiments to test the performance of the framework were first carried out, following which, the PPO method was used to train a policy for executing the search-and-rescue task. The inputs to the neural network included a 50×50 global map matrix, and a 12 dimensional state vector comprising of x, y and ψ of the ANYmal quadruped with respect to the world frame, x, y, z, ϕ, θ and ψ of the quadcopter with respect to the world frame, and a 3 dimensional vector given by $\{GoalFound, Goal_x, Goal_y\}$, where $GoalFound$ is either 0 in case the goal is yet undiscovered, during which $Goal_x$ and $Goal_y$ are both 0, or 1, in the case the goal is discovered and the terms $Goal_x$ and $Goal_y$ then represent the x and y positions of the system goal.

The output of the policy network is a 6 dimensional vector given by $\{\dot{x}_A, \dot{y}_A, \dot{\psi}_A, x_Q, y_Q, z_Q\}$ where A refers to ANYmal and Q refers to the quadcopter.

The reward at each time step, used for the search-and-rescue task, is computed as shown below

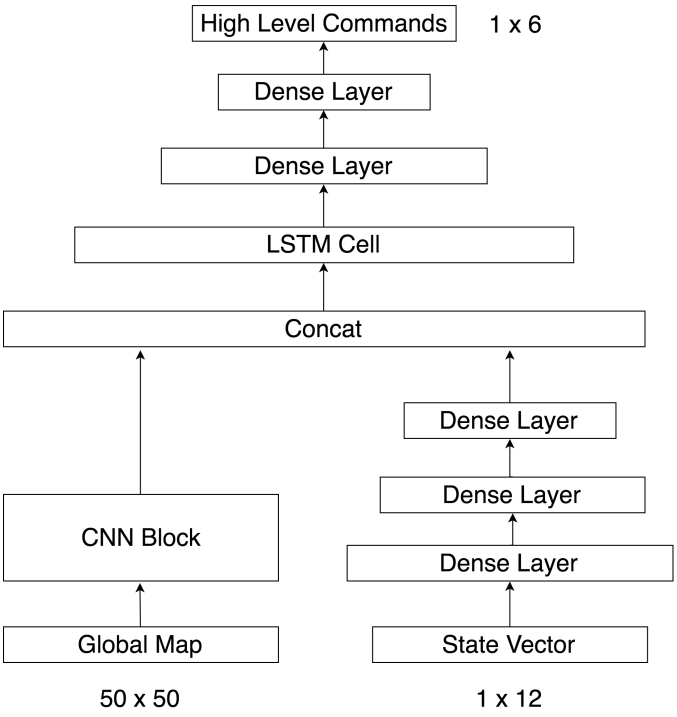

Figure 5. The neural network architecture used for value function and policy estimation using PPO for high level robot control of the heterogeneous swarm system.

Reward, $R \leftarrow 0$
if *ANYmal_AtGoal* == *False* **then**
  if *GoalDiscovered* == *False* **then**
    $R \leftarrow R - c_1$
  else
    if *RewardedForGoalDiscovery* == *False* **then**
      $R \leftarrow R + c_2$
      RewardedForGoalDiscovery $\leftarrow$ *True*
    $R \leftarrow R - c_3 \times$ *Dist_ANYmalFromGoal*
  $R \leftarrow R - c_4 \times$ *Quadcopter_OffAreaOfOperation*
  $R \leftarrow R - c_5 \times$ *ANYmal_OffAreaOfOperation*
  $R \leftarrow R - c_6 \times$ *Quadcopter_DistanceFromDesiredHeight*
  $R \leftarrow R - c_7 \times$ *Collision_QuadcopterWithObstacles*
  $R \leftarrow R - c_8 \times$ *Collision_ANYmalWithObstacles*
  $R \leftarrow R + c_9 \times$ *ExploredRegionsInGlobalMap*

where c_i is a scaling term.

The neural network architecture required considerable amount of tweaking. During some of the initial tests, it was observed that no learning occurred due to extremely low randomness in the initial values of the network parameters. The performance curves are as shown in Figure 6. However, a great improvement was observed upon changing some of the transfer functions and also by using some normalization techniques, as is shown in the Figure 7.

Following some initial tests and having observed improvement in the learning process, the scaling terms in the reward function were accordingly adjusted to obtain faster learning for desired behavior.

Figure 6. No learning was observed during some of the initial tests even after more than 3k policy iterations.

Figure 7. Upon tweaking some of the training parameters, a significant improvement was observed in the learning process.

V. Observations and Results

The trained policy was used to perform task allocation based heterogeneous swarm optimization. It was observed that the PPO strategy failed to learn in the case where the parameters were not tweaked. However, it did a lot better when a few machine learning hacks were used. Moreover, it was observed that the PPO strategy was sample inefficient requiring policy iterations of more than 20k to perform the desired task. Also, the complexity of the neural network used for training further added to the training period. It was observed that setting the quadcopter elevation to be constant significantly improved the performance of the training algorithm.

For high level control policy, the performance observed was satisfactory. The policy managed to perform extremely well when the goal position was near the center of the area of operation (AOP), however, failed at several instances when the goal position was near the edges. This could have been due to the penalty given as the quadcopter moves away from the AOP because of which it prefers exploration around the center of the AOP. The exploration done by the quadcopter was thus limited to the center of the AOP. The high level control of the quadruped, however, was extremely well executed. It managed to reach the goal, while avoiding the obstacles in most of the cases. There were, however, cases when the policy could not generate a feasible path for the quadruped. This happened mostly when several obstacles were very close to the goal.

Unlike in the case of high-level control, the low-level control policy performed extremely well for quadruped locomotion without any known instances of failure. The TRPO algorithm, despite a simple policy network, managed to achieve a locomotion behavior which was faster than the hand-designed controller for the ANYmal quadrupedal robot. Moreover, in some of the other experiments performed to test the TRPO performance for posture recovery, in which case, when the ANYmal was pushed until it tipped off, it was observed that the robot managed to recover to its base state in a lot better manner than in case of a classical-control-theory based approach.

When a general policy was used to perform both, high and low level control of the robotic systems, it failed to learn. However, the use of a hierarchical approach promised a much desired robotic swarm behavior.

VI. Conclusion and Future Work

The report reviewed the performance of the policy gradient techniques for use in heterogeneous robotic swarm optimization. Having observed the behavior for unified high and low level control, it was concluded that the high dimensional problem was not suitable for execution using the learned PPO policy. Instead, the hierarchical architecture promised a lot better performance. Moreover, though not quantitatively analyzed, it was concluded that the policy trained using TRPO for low level control of ANYmal was extremely well suited for the application. For future work, performing a comparative analysis of each of the reinforcement learning and classical-control-theory based algorithms shall be considered.

The work on this project has further motivated development of MRCF which shall be extremely handy especially for rapid prototyping solutions for various desired robotic behaviors. Here, the motivation and approach for development of the framework is detailed.

As demonstrated by the authors of [16] [17] and [18], who have successfully implemented RL strategies for various robotic applications including control of robotic manipulators, helicopter aerobatics, and even quadrupedal locomotion, RL promises significant development in the field of robotics. 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 reward function in order to learn an optimal policy resulting in a sensible robotic behavior. Often, this reward function needs to be tuned by a human expert. For tasks such as quadrupedal navigation through rough terrain, computing a reward function is also significantly more difficult than for tasks such as posture recovery, which when solved using an RL algorithm results in a near-optimal policy.

As a solution to the reward function tuning required in RL problems, the inverse reinforcement learning (IRL) [34] 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.

The authors of [35] and [36] have successfully implemented IRL techniques for perception and control tasks. These methods, along with other RL algorithms, will be included in the MRCF. Moreover, since most of these IRL approaches require an extra step of solving an RL problem upon having recovered a reward function, thereby increasing the training time, an alternative approach of imitation learning (IL) [37], in which an agent is trained to perform a task from demonstrations by learning a mapping between observations and actions, can also be used.

Most of the mentioned algorithms can be used for both low and high level control. However, based on previous experiments and as described next, using classical control theory approaches for low level control with deep RL approaches for high level control enables the development of control solutions that are not attainable by each method alone. It is therefore important to include controllers such as the linear quadratic regulator [38], MPC, and QP in the MRCF. These controllers will also be used in the hierarchical software architecture for solving problems such as heterogeneous swarm optimization.

As previously described, much of the work on RL, IRL and IL has used either physics simulators that often allow non-realistic interactions, for example MuJoCo environments [25] often allow applying forces from a distance, or on simple robotic platforms such as manipulators (e.g. non-changing dynamics). In contrast, less research work has focused in the field of mobile robotics where the control problem tends to be more complex. This is often due to issues such as sensor and actuation uncertainty as described by the authors of [39], dynamically changing environments, which may require control optimization at each time step, and the use of simplified mechanical models as done by authors of [40]. Throughout the work that shall be carried out during development of MRCF, the aim shall focus on developing and extending upon the proposed algorithms by the reinforcement learning research community for real-world application and accordingly plan on realistic benchmarks and frequent real-world trials.

For RL problems where the reward function needs to be well tuned, the policy often converges to a sub-optimal state, making evident the need for solutions based on techniques such as IRL. One project that the author of this report had previously worked on involved extending on the technique used in [41] on *generative adversarial imitation learning* (GAIL) to quadrupedal locomotion. Based on the principles of IRL, 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. The work carried out as part of the project, as shown in Figure 8, on the quadrupedal robot platform, ANYmal [19] can be summarized as; planning sequences of footstep placements for the ANYmal quadruped using elevation map and robot state as observations, to ascend and descend a wide range of stairs using generative adversarial imitation learning by providing expert demonstrations. 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 the 12 Degrees of Freedom (DoFs) of the quadruped resulted in a policy producing not smooth trajectories, and thus, not suitable for robotic control. Therefore, footstep plans were generated using GAIL and a controller based on *FreeGait* [42] was used for low-level control. The experiments carried out as part of this project motivated further research in the development of hierarchical models of robotic control.

Following the work performed as part of this project, the proposed future research work shall involve developing a unified approach to robotic control by combining deep learning and control theory based methods. This shall allow extending upon previous work and developing a versatile robot control framework, usable in a wide range of application domains.

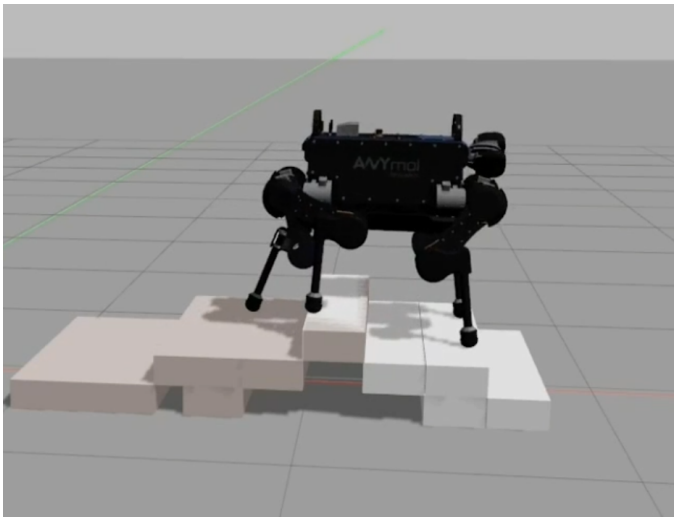

Figure 8. The Gazebo simulation environment for the experiments carried out as part of the GAIL strategies for ascending and descending different kinds of stairs.

The beginning of the research work will focus on the development of MRCF which will consist of several existing robotic control algorithms. Moreover, the MRCF will be compatible with several robotic platforms and the Robot Operating System (ROS). During the development of the framework, each of the added algorithms shall be tested for its performance through comparative analysis with pre-existing frameworks such as the *OpenAI Gym*. The training will be done by building test environments for the *RAI-Sim* and the *Gazebo* simulators.

After having tested the algorithms packaged with MRCF, the project will focus on the development of a heterogeneous swarm environment, consisting of aerial and ground robots. Each of the robots will be controlled by a learned policy at a lower-level. For example, the *ANYmal* quadruped will be controlled by a learned policy that given $\dot{x}$, $\dot{y}$, and $\dot{\psi}$ commands as input, will produce 12 joint torques, as control inputs for the robot's joints. For different robot platforms, however, different control strategies will be used, either based on control theory or deep learning, depending on the platform. One of the main challenges faced during development of a control strategy for a robot is the quantitative analysis of the performance of the algorithm. For this reason, a thorough comparative study of different types of algorithms used in different environments is necessary. Depending on the robot's behavior and the environment of operation, the best suited control strategy will be used for low-level control.

Having trained policies for low-level robotic control, for various kinds of robotic platforms, implementation of a heterogeneous swarm system for the task of exploring the environment of operation and navigating to different goals shall be done. The modularity of MRCF will enable the fast development of such a system. Various algorithms, including policy gradient methods will then be used for swarm optimization, that is,

training a policy to compute high-level commands for the robotic systems in the swarm. A comparative study to test the performance of each of the algorithms used shall then be performed. Though, many of the reinforcement learning algorithms promise convergence to a near-optimal solution, such a problem will need to account for the varying number of robotic platforms used in the swarm. Through some of the previous work, it was observed that the training period for a varying number of robots is significantly higher than for a fixed number, thereby necessitating more compute resources.

To benchmark the algorithms, a comparative analysis of each of the used methods for various performance parameters including training period, convergence reward, generalization of the algorithm by tests in different environments, and training iterations shall be carried. The benchmarks will include test trials in simulation and with the real robots.

In conclusion, the research project detailed in this report was successfully implemented. It further motivated development of the MRCF, the work of which has already begun. Following the development of the framework, a thorough comparative analysis of various training algorithms for the task of heterogeneous swarm optimization will be performed.

REFERENCES

[1] A. Prorok, M. A. Hsieh, and V. Kumar, "Fast redistribution of a swarm of heterogeneous robots," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 249–255.

[2] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[3] A. K. Das, R. Fierro, R. V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *Departmental Papers (MEAM)*, p. 9, 2002.

[4] W. Ren and N. Sorensen, "Distributed coordination architecture for multi-robot formation control," *Robotics and Autonomous Systems*, vol. 56, no. 4, pp. 324–333, 2008.

[5] J. P. Desai, J. P. Ostrowski, and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots," *IEEE transactions on Robotics and Automation*, vol. 17, no. 6, pp. 905–908, 2001.

[6] E. G. Jones, B. Browning, M. B. Dias, B. Argall, M. Veloso, and A. Stentz, "Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 570–575.

[7] M. A. Hsieh, Á. Halász, S. Berman, and V. Kumar, "Biologically inspired redistribution of a swarm of robots among multiple sites," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 121–141, 2008.

[8] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, B. Jung *et al.*, "Adaptive teams of autonomous aerial and ground robots for situational awareness," *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 991–1014, 2007.

[9] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[10] B. Charrow, "Information-theoretic active perception for multi-robot teams," 2015.

[11] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2219–2225.

[12] M. Yogeswaran, S. G. Ponnambalam, and G. Kanagaraj, "Reinforcement learning in swarm-robotics for multi-agent foraging-task domain," in *2013 IEEE Symposium on Swarm Intelligence (SIS)*, April 2013, pp. 15–21.

[13] H. Iima and Y. Kuroe, "Swarm reinforcement learning method for a multi-robot formation problem," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Oct 2013, pp. 2298–2303.

[14] D. Dimitrov, A. Sherikov, and P.-B. Wieber, "A sparse model predictive control formulation for walking motion generation," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2292–2299.

[15] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2589–2594.

[16] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3389–3396.

[17] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*. Springer, 2006, pp. 363–372.

[18] 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. IEEE, 2004, pp. 2619–2624.

[19] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[20] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, 2017.

[21] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, April 2018.

[22] N. P. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator." in *IROS*, vol. 4. Citeseer, 2004, pp. 2149–2154.

[23] E. Coumans, "Bullet physics simulation," in *SIGGRAPH Courses*, 2015.

[24] R. Smith *et al.*, "Open dynamics engine, 2008," *URL: http://www. ode. org*, vol. 5, 2006.

[25] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

[26] S. Ivaldi, V. Padois, and F. Nori, "Tools for dynamics simulation of robots: a survey based on user feedback," *arXiv preprint arXiv:1402.7050*, 2014.

[27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[28] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[29] S. Wright and J. Nocedal, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[30] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *IN PROC. 19TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, 2002, pp. 267–274.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[33] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, 2011.

[34] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[35] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[36] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[37] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[38] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[39] R. Luo, M.-H. Lin, and R. Scherp, "The issues and approaches of a robot multi-sensor integration," in *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, vol. 4. IEEE, 1987, pp. 1941–1946.

[40] H. Kimura, I. Shimoyama, and H. Miura, "Dynamics in the dynamic walk of a quadruped robot," *Advanced Robotics*, vol. 4, no. 3, pp. 283–301, 1989.

[41] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 4565–4573.

[42] P. Fankhauser, C. D. Bellicoso, C. Gehring, R. Dubé, A. Gawel, and M. Hutter, "Free gait—an architecture for the versatile control of legged robots," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 1052–1058.