Acknowledgement

We would like to acknowledge the Fung Institute for Engineering Leadership, as well as the Artificial Intelligence and Biomimetic Millisystems labs for the means to accomplish this project. Special thanks to Fernando Garcia Bermudez for his guidance and help.

Study of Reinforcement Learning Methods to Enable Automatic Tuning of State of The Art Legged Robots

By

Zihong Lian

A project report submitted in partial satisfaction of the

requirements for the degree of

Master of Engineering

with area of concentration in

Robotics & Embedded Software

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel

Professor Ronald S. Fearing

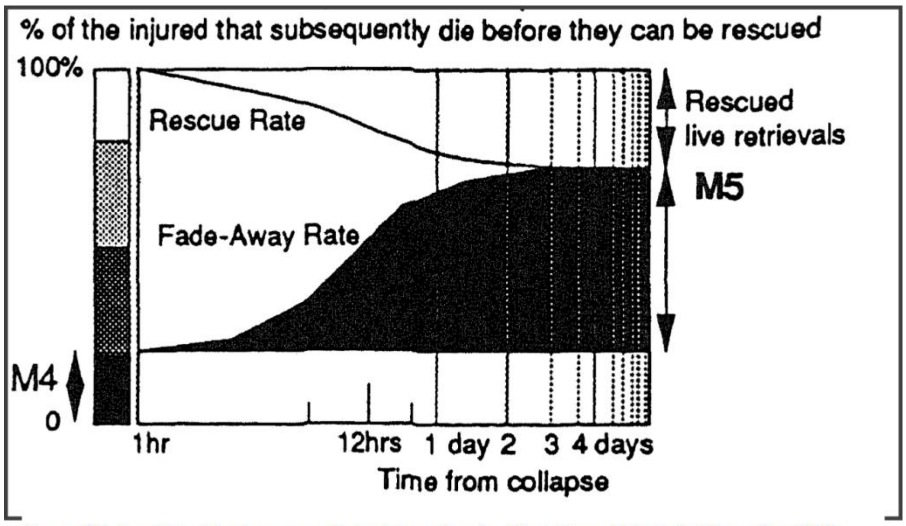
Spring 2012

Abstract

Search and rescue is often a slow process, which puts people at risk of being trapped, stranded or even worse, killed during natural disasters, such as earthquakes, floods and hurricanes. In order to provide better rescue assistance and to achieve high survival rates, we need efficient, cost effective and small crawling robots to execute search and rescue operations during disaster situations, especially in reaching spaces that are inaccessible for larger robots or are harmful to rescuers. Thus, I worked on improving the walking speed and autonomous behaviour of OctoRoACH, an inexpensive and robust palm-sized eight legged robot developed by the Biomimetic Millisystems Lab, together with my capstone project members and advisors at UC Berkeley. Our results show that reinforcement learning algorithms is useful to improve the walking speed of existing search and rescue robots across different terrains and save more lives during disaster situations.

Introduction

The world population reached 7 billion in October 2011, with the majority concentrated in underdeveloped countries. This means that a huge amount of people are vulnerable to natural disasters, such as earthquakes, floods and hurricanes.



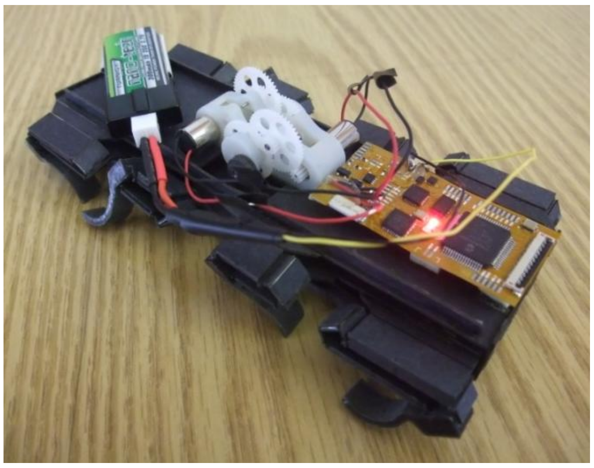
Source: "Factors determining human casualty levels in earthquakes: Mortality prediction in building collapse"[16]

Fig. I1. % of the injured that subsequently die before they can be rescued. [16]

During disaster situations, the goal of search and rescue operations is to rescue the greatest number of people within the shortest time possible, while reduce potential risks to the rescuers. As implied in the graph above, the more efficient search and rescue operations are executed, the less time victims spend trapped in rubble and the higher their chances of survival are. Unfortunately, search and rescue is often a slow process, which puts people at risk of being trapped, stranded or even worse, killed. Some of the main reasons of the lack of efficiency in the current search and rescue operations are the physical constraints of humans, as well as the desire to maintain the safety of rescue personnel.

As such, in order to provide better rescue assistance and to achieve high survival rates, we need efficient, cost effective and small crawling robots to execute search and rescue operations during disaster situations. State of the art legged robots, such as the OctoRoACH which are developed by the Fearing lab at UC Berkeley, have been demonstrated to be capable to locomote across a variety of challenging terrains. However, their efficiency of locomotion can be significantly affected by the timing of the legs. Tuning the timing parameters by hand can be time-consuming and the best setting for the parameters can vary with wear of the robot and when terrain properties change.

Therefore, to provide solutions for the challenges stated above, my teammates and I am currently working on OctoRoACH, an inexpensive, light and robust palm-sized eight legged robot developed by the Biomimetic Millisystems Lab at UC Berkeley, in collaboration with my capstone project team members, teacher advisors at UC Berkeley. This self-contained robot is also developed for Micro Autonomous Systems and Technology (MAST) for reconnaissance missions. Our objective is to study reinforcement learning methods to enable automatic tuning of the control policy parameters. This will enable legged robots, such as OctoRoACH to more efficiently traverse wide varieties of terrains, such as wood, carpet, gravel, grass, sand, and etc.

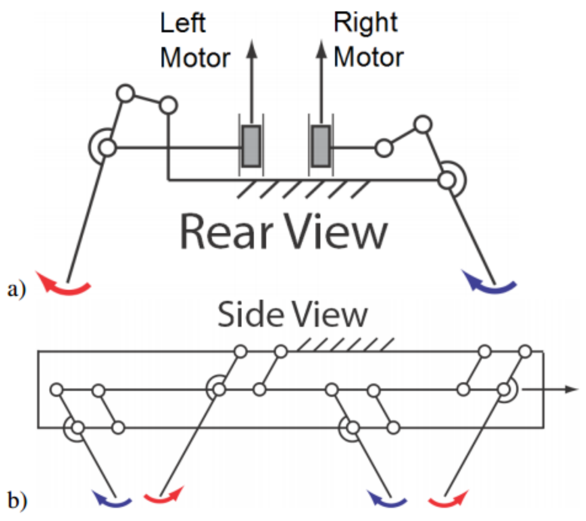


Throughout this report, we will provide you with better understanding of our study and present you our detailed results. present results and conclusions of our findings. Firstly, we will establish the context of existing bio-inspired legged robots and study various existing machine learning algorithms that improve the walking speeds of OctoRoACH. Secondly, we will describe the methodologies we used to implement and evaluate the different types of reinforcement learning algorithms on both simulated and physical OctoRoACH. Thirdly, we will discuss the results of our implementations and evaluations. Fourthly, we will provide conclusions and recommendation of the different reinforcement learning algorithms that we have tested and evaluated thus far. We will also discuss potential future work.

Context / Literature Review

It is time-consuming when hand tuning the timing parameters and that the best parameters setting of each individual robot is dependent upon the robot condition and the change in terrain properties. In order to overcome this challenge, this project requires the study of reinforcement learning (RL) methods to enable automatic tuning of the control policy parameters. This will enable these robots to traverse wide range of terrains more efficiently.

Fig. LR1. [10]

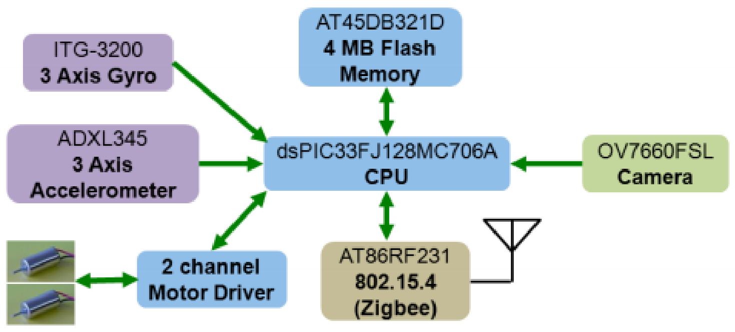


Ideal robot kinematics demonstrating the kinematic coupling enabling fore-aft and in-out motion of legs. The two sides of the robot are driven independently.

a) Rear view of the ideal robot kinematics: ab- and adduction of leg pairs on each side occurs out of phase when the middle member of the linkage is translated vertically.

b) Side view of the ideal robot kinematics: protraction and retraction of the legs is controlled by motion of the middle member in the fore-aft direction. The alternate pairs move approximately 180 degrees out of phase with each other.

Fig. LR2. [10]



Block Diagram of ImageProc2.2 Controller Board.

DynaRoACH is a bio-inspired design, underactuated hexapedal robot prior to the eight-legged OctoRoACH. DynaRoACH is driven by a single actuator to produce a high speed alternating tripod gait and run at 14 body lengths per second, whereas OctoRoACH has two actuators that are driven independently (see Fig. I1.). The maximum speed and stride frequency for DynaRoACH is 1.4 m/s and 20 Hz respectively and for OctoRoACH are 0.5 m/s and 25 Hz respectively. DynaRoACH achieves dynamic turning by modulating the front or middle leg stiffness, while instead OctoRoACH uses gyro feedback in a closed-loop steering system to control its turning using PID (see Fig. I2). Both robots have real-time measurements of the motor back EMF. The advantage of OctoRoACH is that it can use leg velocity control and rate-gyro-based heading control system to steer its direction or turning rate efficiently. However, the disadvantage of OctoRoACH is that it is an early design and lacks stability due to inconsistencies with the mechanical tuning of leg and body compliance, and is affected by terrain properties (i.e. surface friction, irregular height and hardness). In conclusion, the use of differential steering and rate gyro are useful for searching the best possible control policy parameters. [1][10]

The quadrupedal "LittleDog" robot uses a hierarchical control architecture to traverse over challenging terrains. The controller consists of (i) high-level planner that plans a set of footsteps across the terrain, (ii) low-level planner that plans trajectories for the robot's footsteps and center of gravity (COG), and (iii) low-level controller that tracks these desired trajectories using a set of closed-loop recovery mechanisms to keep track of preferred trajectories and improve system performance. This paper also discussed a motion capture (MOCAP) system that estimates the position and orientation of the robot's body and joint angles by tracking reflective markers attached to the robot. The advantage of the hierarchical control system is that it enables LittleDog to plan a sequence of joint angles and steadily traverse rough terrains and climb over obstacles nearly as tall as the robot's legs by applying relevant control inputs to achieve the desired trajectory. However, the disadvantages of this approach are (d.i) moving the COG more to achieve greater stability may increase the likelihood of collision between LittleDog's legs and the terrain and (d.ii) not applicable to OctoRoACH as it does not have a hierarchical control architecture. In conclusion, MOCAP is useful in helping us collect data on OctoRoACH and the process will be detailed in the methodology section. [2]

The first controller capable of omnidirectional path following with parameters optimized simultaneously for all directions of motion and turning rates on LittleDog is presented in another paper. Although challenging, a simulator with a Reduced Rank Regression (RRR) dimensional reduction algorithm can be used to identify a low-dimensional subspace of policies that spans the variations in model dynamics to optimize controller parameters (i.e. omnidirectional path following and turning rates). MOCAP was used to track reflective markers on LittleDog and obtain its state estimation (i.e. direction and turning angles). This concept was applied to OctoRoACH when timestamps, positional trajectories and rotational matrices were obtained to calculate its traversing speed and accuracy for a range of control policy parameters. The advantage of RRR is that it can be implemented to OctoRoACH, in which the control policy parameters can be reduced to perturbation of left and right motor thrusts, travel time and PID controller values. However, the disadvantage of RRR is that the reduced dimensional policies may exclude essential parameters to represent the model of a robot. In conclusion, low-dimensional control policy parameters can be defined for OctoRoACH in search for the best possible gait parameters. [3]

A policy search method for a Markov decision process (MDP) or partially observable MDP (POMDP) is based on the observation that POMDP can be transformed into an “equivalent” one and only POMDP with deterministic transitions needs to be considered and further estimated. The search for a policy with the best possible estimated parameters can be achieved with the optimized estimate from a policy search method initiated by user specified criteria, provided these estimates will be uniformly good. The advantage of using POMDPs for control is that broadly speaking they are quite robust to variations in the transition and observation probabilities [12]. The disadvantage of policy search is that the convergence property of the optimization procedure is affected each time the objective function changes [13]. In conclusion, the policy search method can be implemented on the OctoRoACH by randomly perturbing a deterministic set of control policy parameters (i.e. left and right motor thrusts) and optimize the estimated values to achieve the best possible parameters. [7][8]

Locomotion of legged robots is a challenging multi-dimensional control problem as the coordination of motions in all legs of the robots is needed while considering stability and surface friction. Also, the lack of accurate simulators force researchers to perform the learning entirely on actual robots, which is simpler but requires more human time and intervention, and poses challenges (i.e. sparse training data, dynamical complexity, and discrepancy between the ideal and real locus). Policy gradient RL algorithm, which was implemented on actual quadrupedal Sony Aibo robots, automatically searches for an N-dimensional parameterized walk while optimizing for both forward gait speed and stability, using a multi-criteria objective function that includes acceleration, gait speed and stability. Enabling a robot to autonomously learn to improve its own performance, RL automatically estimates the gradient of an initial set of randomly perturbed parameters and reaches a local optimum to search for parameters of the fastest possible walk. The advantages of this approach over hand-tuned methods are (a.i) resulting gait is reasonably faster and steadier than previous hand-tuned and learned solutions on the same robot platform, (a.ii) find a fast gait efficiently, (a.iii) less human bias and intervention, (a.iv) easily applied to various surfaces and robots, and (a.v) robot’s visual object recognition is greatly improved. However, the disadvantages of this approach are (d.i) RL algorithms may generate some intermediate exploratory gaits that cause physical damage to the robots over time and lead to inconsistencies between them, which will need further fine-tuning to improve individual robot performances, (d.ii) evaluations on actual robots are noisy and take a long time than that in robot simulations, (d.iii) starting point for the search could affect final results, (d.iv) gradient calculations cannot be precise and empirical estimation of the gradient by sampling can be computationally expensive due to the large search space and the temporal cost of each evaluation, (d.v) Aibos lack sensors that can be used during training to provide controller closed loop feedback. But in our project, OctoRoACH's gyro sensor data and the use of MOCAP can provide us with closed loop feedback. In conclusion, there is a need for the policy gradient RL algorithm to be implemented on the robot to efficiently improve gait locomotion efficiency. [4][5][6][8]

Likelihood ratio policy gradient RL methods can be derived from an importance sampling perspective and have been some of the most successful RL algorithms, especially on physical systems. A full estimation of the expected return function is provided and global search over the importance sampled expected return gradient information can be used to achieve faster learning. Likelihood ratio methods (i) use past experience to estimate only the gradient of the expected return U at the current policy parameterization, rather than to obtain a more complete estimate of U and (ii) use past experience under the current policy only, rather than using all past experience to improve the estimates. The advantages of likelihood ratio methods are (a.i) theoretically faster convergence rate, (a.ii) gradient estimation process is no longer threatened by the complex control of these variables, since the generation of policy parameter variations is not needed. (a.iii) produced the most real-world robotics results and is guaranteed to obtain the fastest error convergence for a stochastic system. The disadvantages of likelihood ratio methods are (d.i) when used with a deterministic policy, a system model has to be maintained, which can be difficult to obtain for continuous states and actions. In conclusion, importance sampling based methods will be tested on OctoRoACH as they outperform non-importance sampling based algorithms. [9] [11]

In general, the different RL algorithms will need to be implemented, compared and evaluated before the optimal control policy parameters can be determined. The experiments will initially be carried out manually before the automated RL process takes place.

Methodology / Approach

Reinforcement learning methods to enable automatic tuning of the control policy parameters were studied to enable OctoRoACH to traverse wide range of terrains more efficiently. The team consists of three students who were from a varied specialized educational background, such as robotics, software engineering and computer engineering. Fortunately, all three had completed at least an undergraduate degree and all had basic technical mastery that ranges from hardware to software. None of us have studied reinforcement learning methods prior to this study.

As proof of concept, the team was required to simulate OctoRoACH to evaluate RL methods that may potentially be used to optimize the gaits of an actual robot in the next semester. The team chose Blender 3D that works on top of Bullet physics engine, but faced too many technical difficulties, such as steep learning curve and immature physics application features.

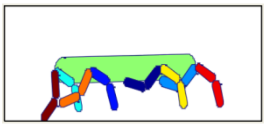
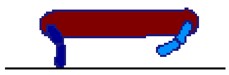
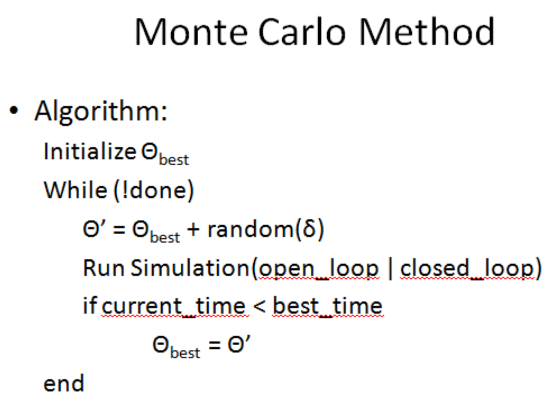


Fig. M1. Two and eight-legged robot simulations.


Eventually, an eight and two legged robot was simulated in a 2D physics simulation environment using MATLAB, namely CapSim (see Fig. M1.). Algorithms that were implemented were Monte Carlo, convex optimization, and policy gradient RL algorithms.

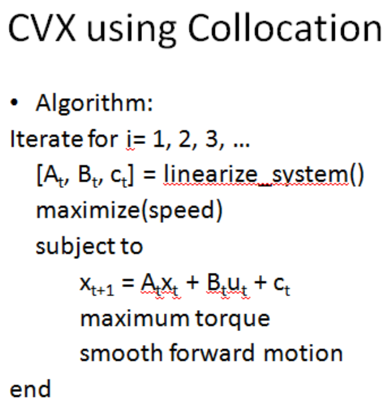


Monte Carlo Method

- Algorithm:

```
Initialize $\Theta_{best}$
While (!done)
    $\Theta' = \Theta_{best}$ + random($\delta$)
    Run Simulation(open_loop | closed_loop)
    if current_time < best_time
        $\Theta_{best} = \Theta'$
end
```

Fig. M2. Monte Carlo Algorithm



CVX using Collocation

- Algorithm:

```
Iterate for i= 1, 2, 3, ...
    [$A_t$, $B_t$, $c_t$] = linearize_system()
    maximize(speed)
    subject to
        $x_{t+1} = A_t x_t + B_t u_t + c_t$
        maximum torque
        smooth forward motion
end
```

Fig. M3. CVX Collocation Method

```
$\pi \leftarrow InitialPolicy$
while !done do
    $\{R_1, R_2, \dots, R_t\} = t$ random perturbations of $\pi$
    evaluate( $\{R_1, R_2, \dots, R_t\}$ )
    for $n = 1$ to $N$ do
        $Avg_{+\epsilon,n} \leftarrow$ average score for all $R_i$ that have
                a positive perturbation in dimension $n$
        $Avg_{+0,n} \leftarrow$ average score for all $R_i$ that have a zero
                perturbation in dimension $n$
        $Avg_{-\epsilon,n} \leftarrow$ average score for all $R_i$ that have a
                negative perturbation in dimension $n$
        if $Avg_{+0,n} > Avg_{+\epsilon,n}$ and $Avg_{+0,n} > Avg_{-\epsilon,n}$ then
            $A_n \leftarrow 0$
        else
            $A_n \leftarrow Avg_{+\epsilon,n} - Avg_{-\epsilon,n}$
        end if
    end for
    $A \leftarrow \frac{A}{|A|} * \eta$
    $\pi \leftarrow \pi + A$
end while
```

Pseudocode for the N-dimensional policy gradient algorithm.

During each iteration of the main loop t policies are sampled near π to estimate the gradient around π, then π is moved by an amount of η in the most favorable direction.

Fig. M4. Pseudocode for N-dimensional policy gradient algorithm[4]

The team implemented these methods and modeled a virtual robot as close as possible to the specifications of the actual OctoRoACH. The algorithms were evaluated and the results show that policy gradient RL method enables the simulated robot to walk at a higher speed as compared to the other methods, as shown in the plot below:

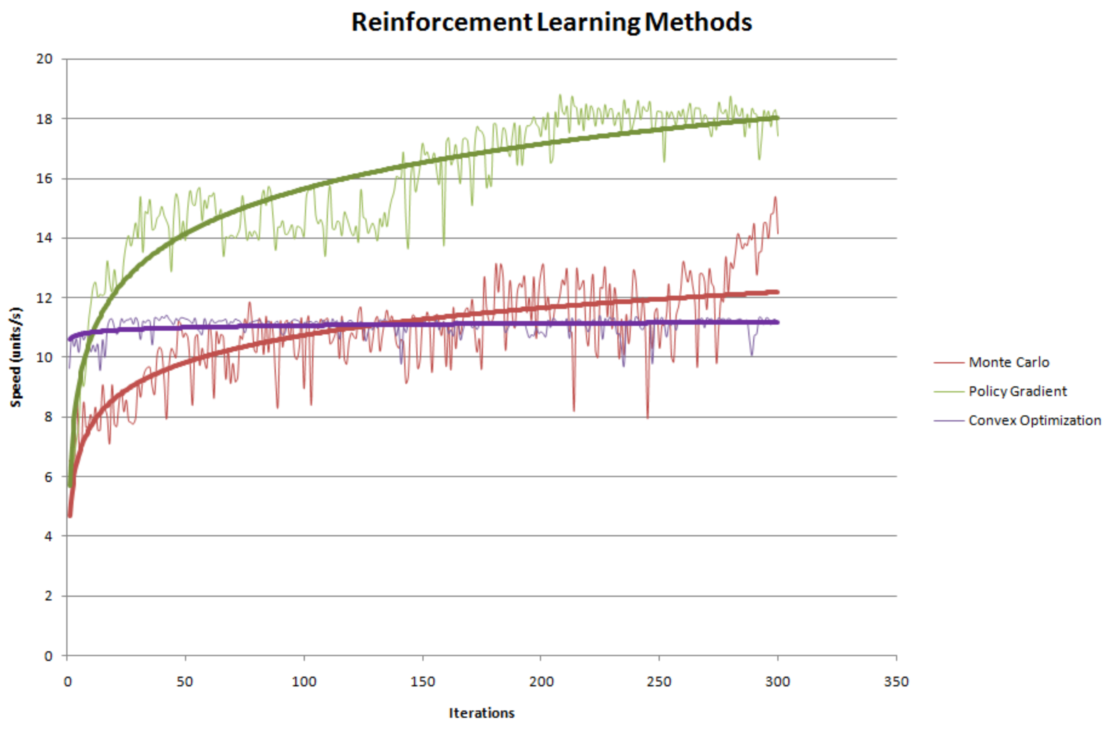
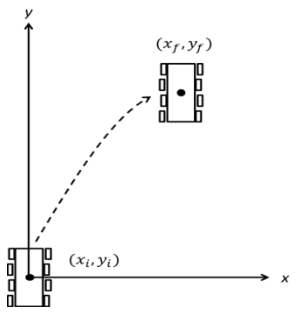


Fig. M5. Reinforcement Learning Methods, i.e. Monte Carlo, Policy Gradient & CVX.

The disadvantages of these simulations are (i) difficult to apply to actual robots, because they are quite inaccurate as it is only in two dimensions. (ii) not realistic to simulate real-life physical constraints such as gravity, friction, collision, rotation, etc.

The actual OctoRoACH comes with electronic boards that provide differential velocity steering controls, which allows us to tweak left and right motor thrusts to steer its movements. In order to show that RL methods are necessary, experiments such as the open-loop and closed-loop steering control were applied on the OctoRoACH. For open-loop system, several terrains such as wood, carpet and gravel were selected. For each terrain, we used Euclidean distance to obtain the initial positions (x_i, y_i) and final positions (x_f, y_f). Then, open-loop policies were hand-tuned (i) on intended surfaces, (ii) on different surfaces, (iii) for the same surface, compare policies across robots (iv) over time to determine if the physical wear of robot affects policies. The metric used for measuring the performance of the robots are normalized speed and accuracy. The mathematical formulation is shown as follows:



$$Speed = \frac{\sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}}{t}$$

$$Accuracy = \sqrt{\frac{(y_f - y_i)^2}{(x_f - x_i)^2 + (y_f - y_i)^2}}$$

Fig. M6. Initial & Final Position & Trajectories. *Fig. M7. Normalized Speed & Accuracy*

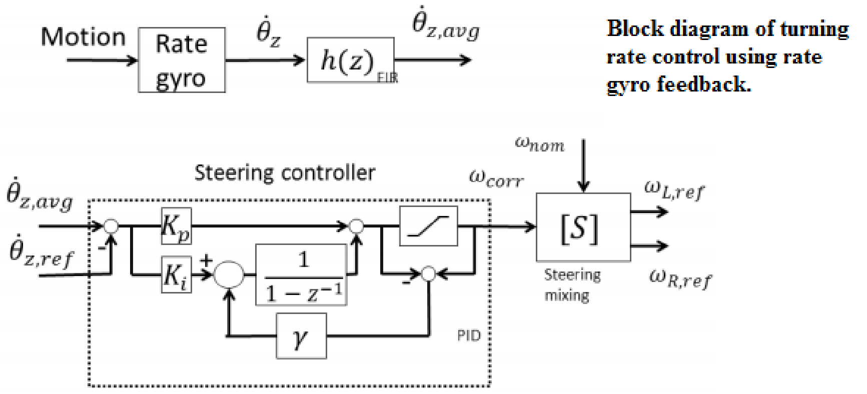


Fig. M8. Closed-Loop Rate Control Gyro Feedback System

For closed-loop steering control, the team had to analyze and use readings from the on board rate gyro feedback data (see Fig. M8) and perform some hand-tuning to allow the robot to increase or decrease right motor thrust and steer its way back to a straight path, if it deviates from its original path caused by the left motor thrust (see Fig. M9). The closed-loop policies were then implemented on intended and different surfaces. Results and plots can be seen in the discussion section. Although less tedious and time-consuming than that of open-loop policy, the team still had to spend time carrying out experiments, such as manually hand tune the closed-loop control policy parameters and tuning the parameters using the on board proportional-integral-derivative (PID) controller. It is shown that reinforcement learning algorithms are needed to improve the gait efficiency and effectiveness.

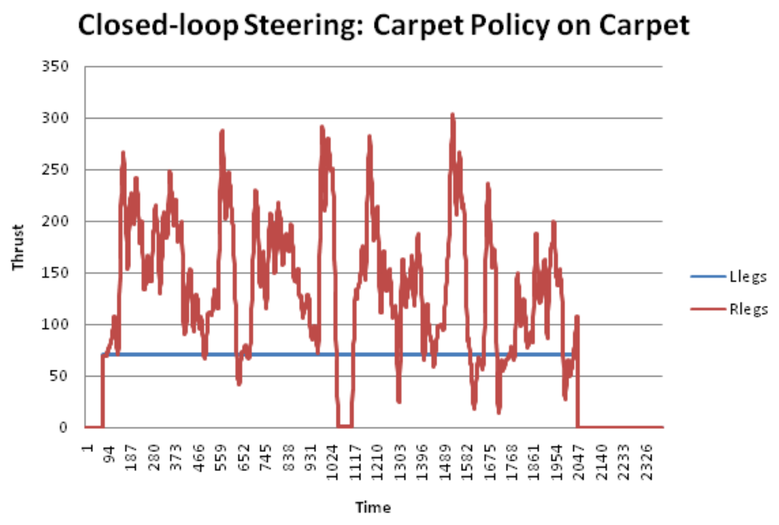


Fig. M9. Plot of Sample Closed-Loop Steering With Varied Motor Thrusts.

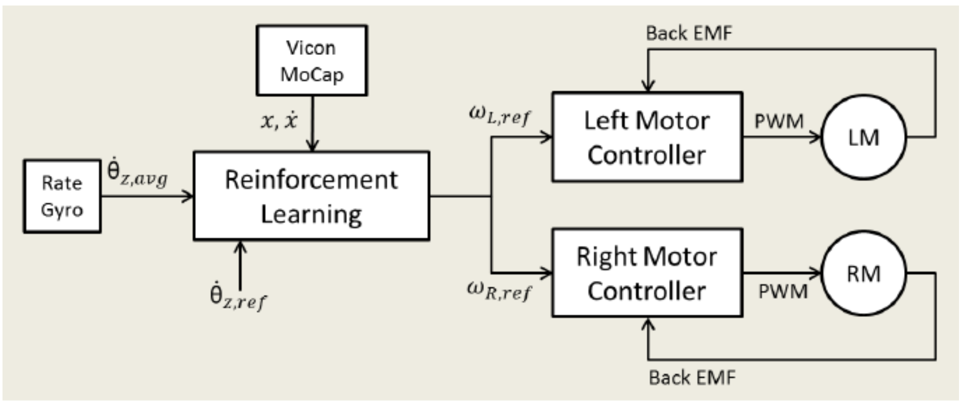


Fig. M10. **Block Diagram of Reinforcement Learning Method on OctoRoACH's Motor Controllers**

RL method is performed with a combination of differential velocity steering, rate gyro feedback, Vicon motion capture (MoCap) feedback, and back EMF (see Fig. M10.). In achieving this goal, reflective markers were placed on OctoRoACH to enable MoCap to detect and save the open-loop trajectories and time stamps of the OctoRoACH for both carpet and wood. Results and plots are presented in the discussion section (see Fig. D8. and Fig. D9.)

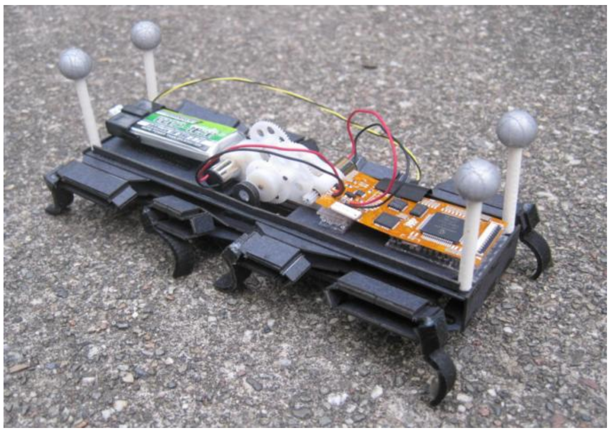
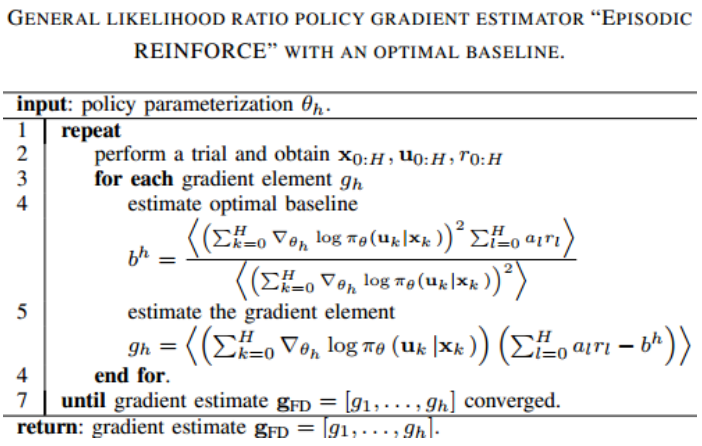


Fig. M11. OctoRoACH with Reflective Markers.

In running the likelihood ratio policy gradient RL experiments, we chose to run our tests on a single surface and for that we chose carpet. Using Python libraries made available to our project on OctoRoACH, we then proceeded to write the RL controller. A common reward function, harmonic mean, R was chosen to optimize both speed and accuracy metrics. The harmonic mean is used as the reward function, R such that

$$\text{Reward function, R} = \frac{2 * Speed * Accuracy}{(Speed + Accuracy)}$$

This decision was made so that a high speed was not chosen at the cost of a very low average, and vice-versa. In order to minimize the impact of noise, six individual measurements for each gradient calculation were taken, and estimated using the least squares method. However, this RL method may not be best suited for an OctoRoACH policy controller, because (i) they do not properly accommodate for mechanical failure or wear, and (ii) representation of physical system by the cost function may be inadequate.


GENERAL LIKELIHOOD RATIO POLICY GRADIENT ESTIMATOR "EPISODIC REINFORCE" WITH AN OPTIMAL BASELINE.

```
input: policy parameterization $\theta_h$.
1 repeat
2     perform a trial and obtain $\mathbf{x}_{0:H}, \mathbf{u}_{0:H}, r_{0:H}$
3     for each gradient element $g_h$
4         estimate optimal baseline
          $b^h = \frac{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\right)^2 \sum_{l=0}^{H} a_l r_l \right\rangle}{\left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\right)^2 \right\rangle}$
5         estimate the gradient element
          $g_h = \left\langle \left(\sum_{k=0}^{H} \nabla_{\theta_h} \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\right) \left(\sum_{l=0}^{H} a_l r_l - b^h\right) \right\rangle$
4     end for.
7 until gradient estimate $\mathbf{g}_{FD} = [g_1, \ldots, g_h]$ converged.
return: gradient estimate $\mathbf{g}_{FD} = [g_1, \ldots, g_h]$.
```

Discussion

The results and plots that were obtained while experimenting with OctoRoACH will be discussed below. Specifically, the open loop steering, closed loop steering with rate gyro feedback system and policy gradient will be discussed.

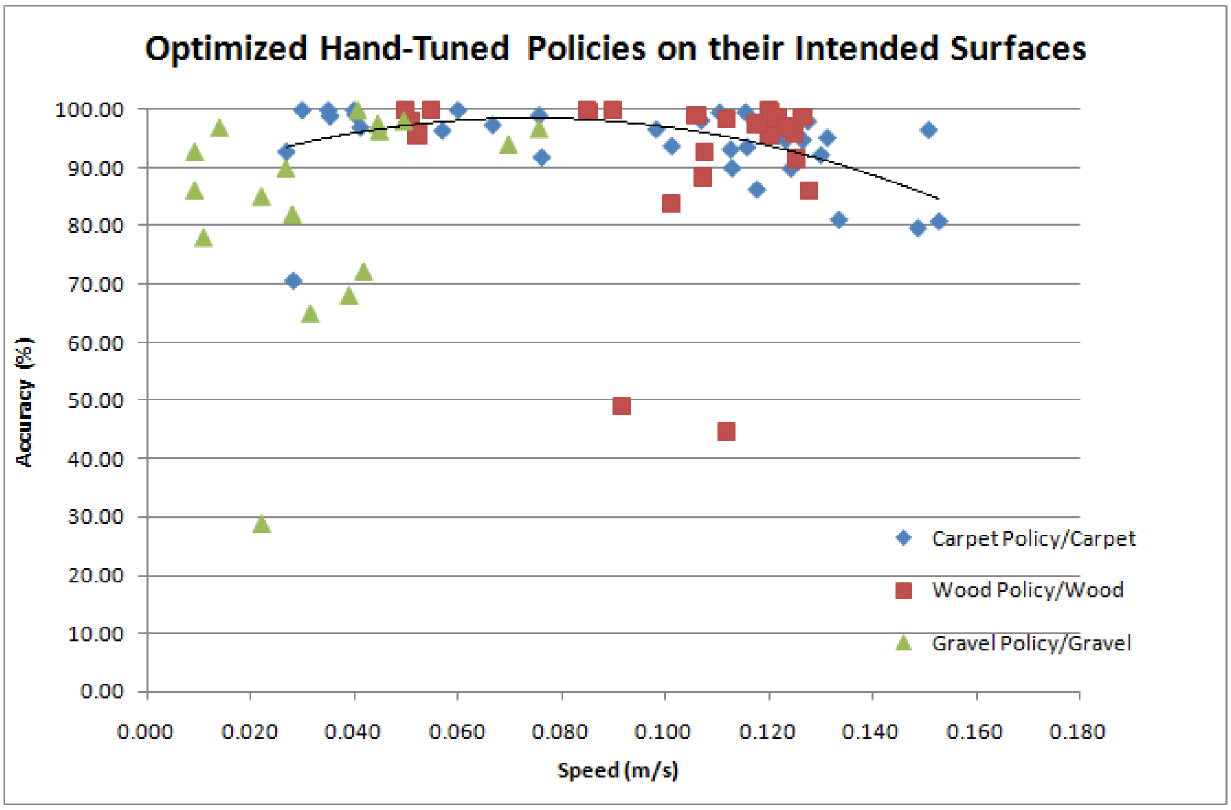


Fig. D1. Optimized hand-tuned policies on intended surfaces, i.e. carpet policies on carpet, wood policies on wood, and gravel policies on gravel.

Hand-tuned open-loop policies can be time consuming yet possibly produce high speeds and accuracies on their intended surfaces. However, when optimal policies were applied on a different surface that was not intended, the performance of robot decreases. Also, since speed and accuracy are inversely related, there is a trade off between speed and accuracy.

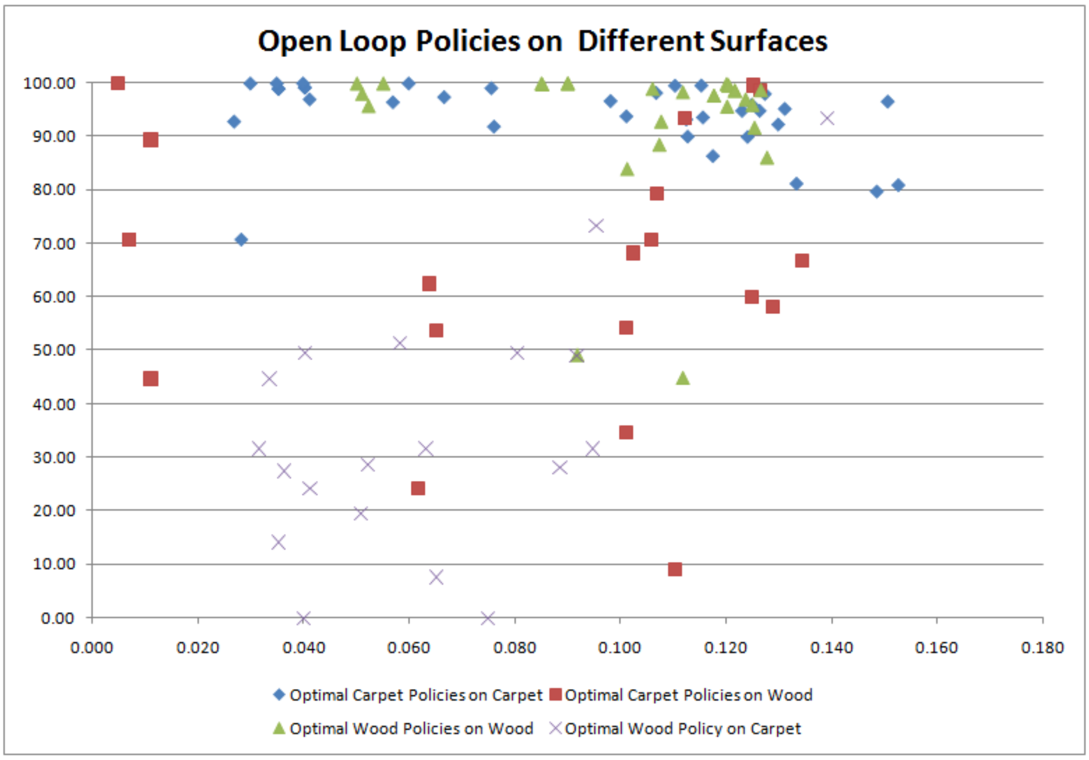


Fig. D2. Open Loop Policies on Different Surfaces, i.e. carpet policies on carpet, carpet policies on wood, wood policies on wood, and wood policies on carpet.

An optimal control policy on a surface does not work on a different surface. Thus, it is obvious that hand tuned policies do not work well on different basic terrains, let alone unpredictable and risky terrains during search and rescue operations in disaster situations.

Policy tuned for wood and tested on 3 different surfaces, i.e. wood, gravel and carpet.

For example, a wood policy that allowed OctoRoACH to walk straight was tuned on a wood surface. However, as shown in the previous graph and the images above, the accuracy and speed performances of OctoRoACH will most probably perform worse than the optimal wood policy. Depending on the mechanical structure of OctoRoACH, the robot may deviate from a straight line, and even worse walk in circles.

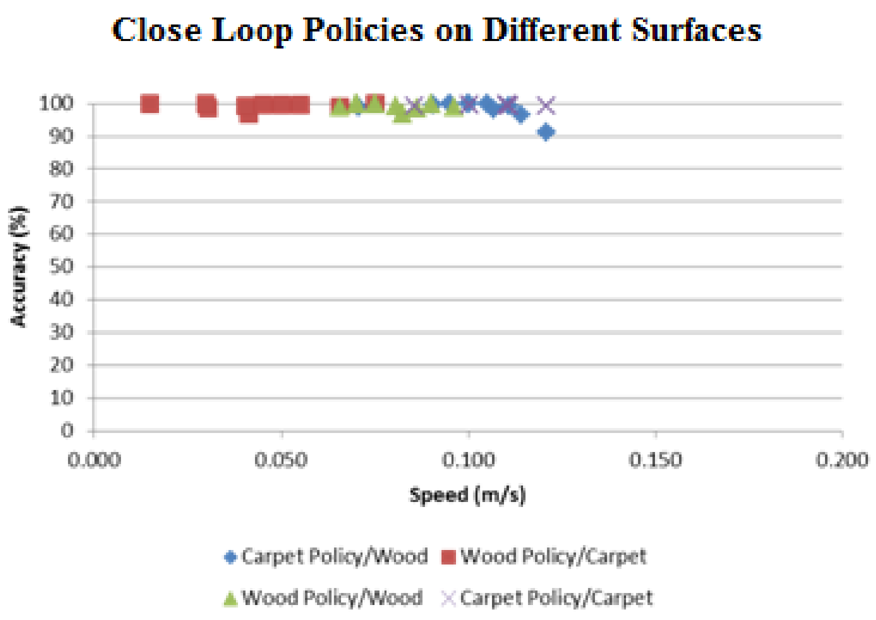


Fig. D3. Closed Loop Policies on Different Surfaces, i.e. carpet policies on carpet, carpet policies on wood, wood policies on wood, and wood policies on carpet.

Hand-tuned closed-loop policies performed well on surfaces they were intended for tuning. As shown in the plot above, closed-loop policies did not improve speed, but increased accuracies.

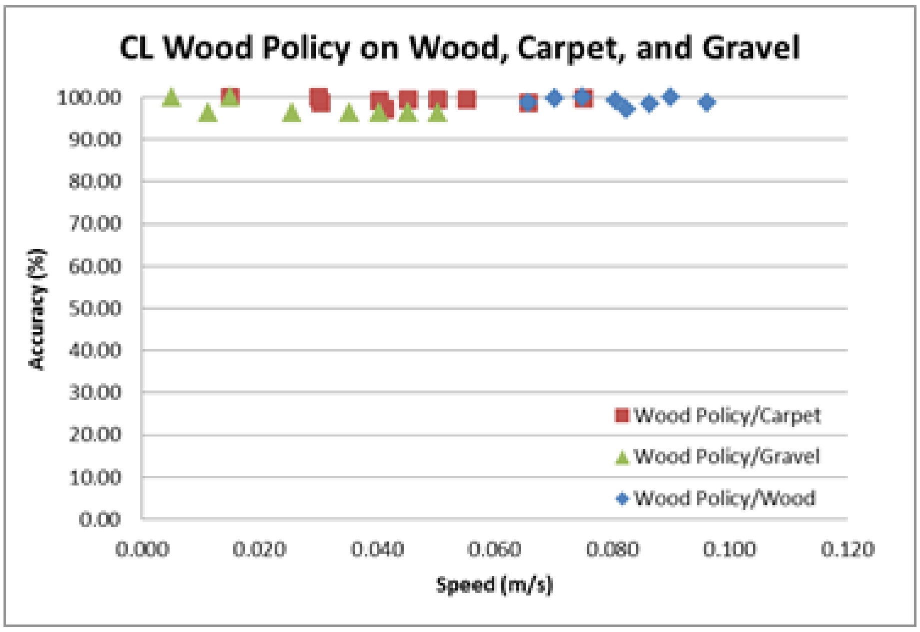


Fig. D4. Closed Loop Wood Policies on Surfaces such as wood, carpet and gravel.

The closed loop wood policies were extended to gravel surface and the results obtained matches our hypotheses closed-loop policies increases accuracies, but not speed.

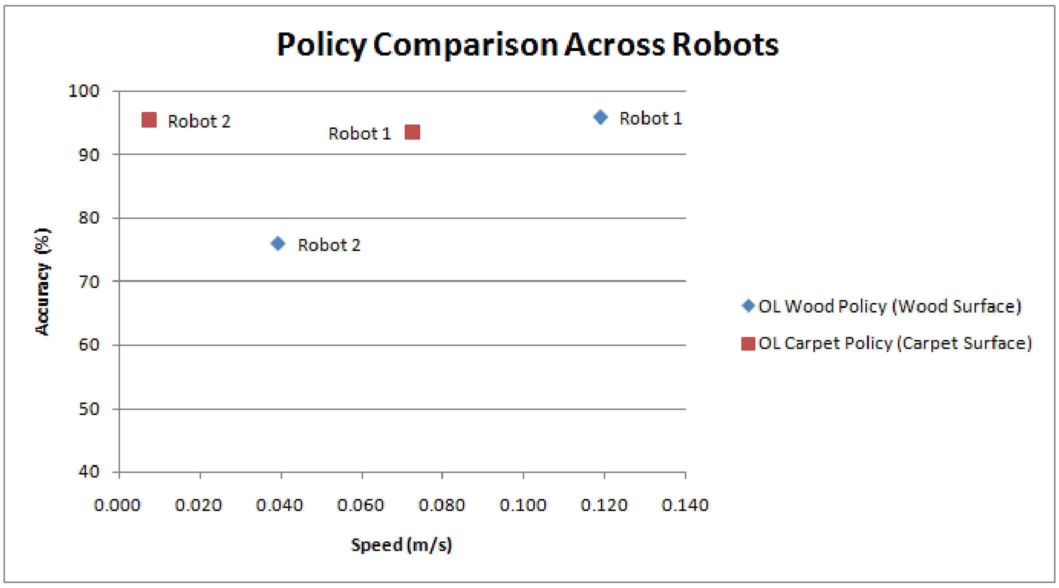


Fig. D5. Policies of one surface, i.e. wood and carpet, are compared across robots.

Hand tuned policies for one robot may not be applicable to another robot. This is most likely caused by mechanical inconsistencies in various aspects, including the leg stiffness, asynchronous leg motions that cause different friction applied on the ground, and etc.

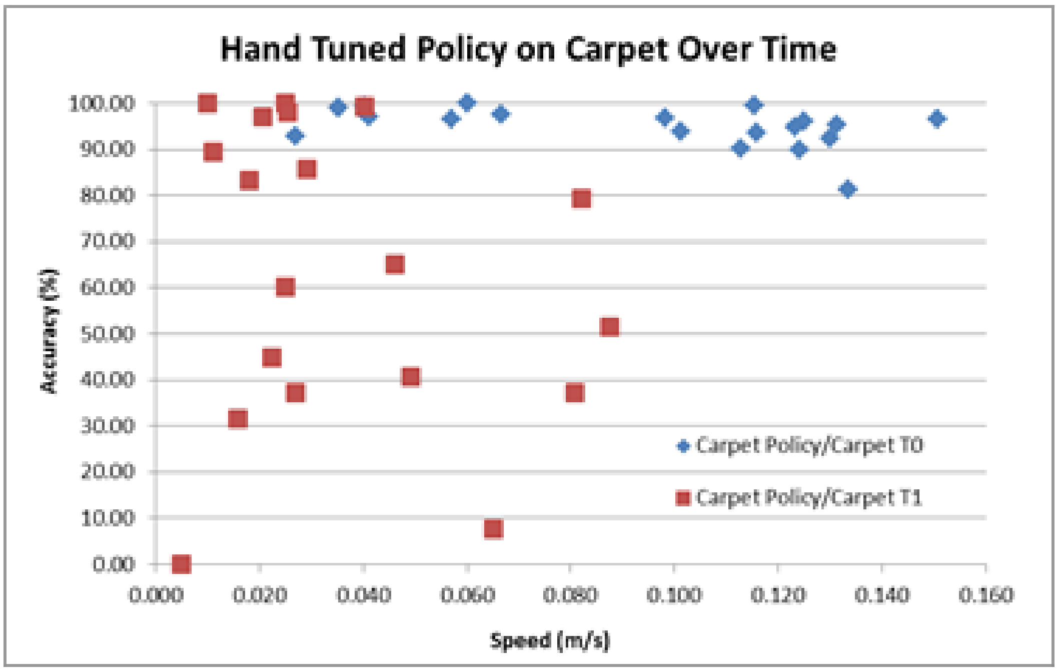
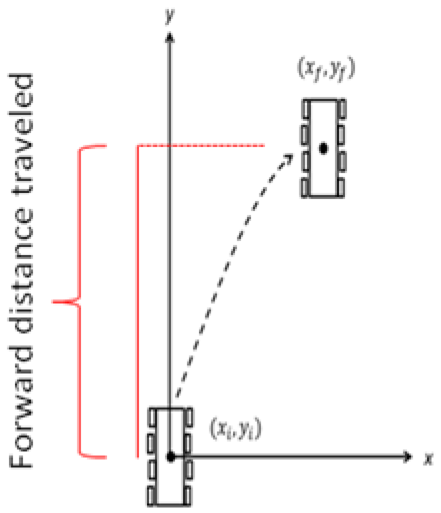


Fig. D6. Hand tuned policy on carpet deteriorates over time.

Due to the physical wear of several parts of the robot, control policy that worked at some point in the past may not work well later. For example, an optimal control policy for carpet that allows the robot to walk on carpet in a straight line initially may result in the robot moving in a circular fashion three hours later. These conditions deteriorate over time and cause more severe mechanical inconsistencies.



Recordings were done to obtain forward trajectories and distance traveled. The initial and final x, y coordinates are taken note of. The traveling speed is then calculated.

Fig. D7.

Due to lack of understanding of the RL method, open-loop experiments were done on wood and carpet surfaces to obtain trajectories. MoCap was used to further characterize the responses to different inputs. A total of 400 tests were run. 200 on each surface, evaluating nearly 100 different thrust settings twice on each surface. Based on the collected data, the forward distance travelled by the robot with several different thrust values were plotted:

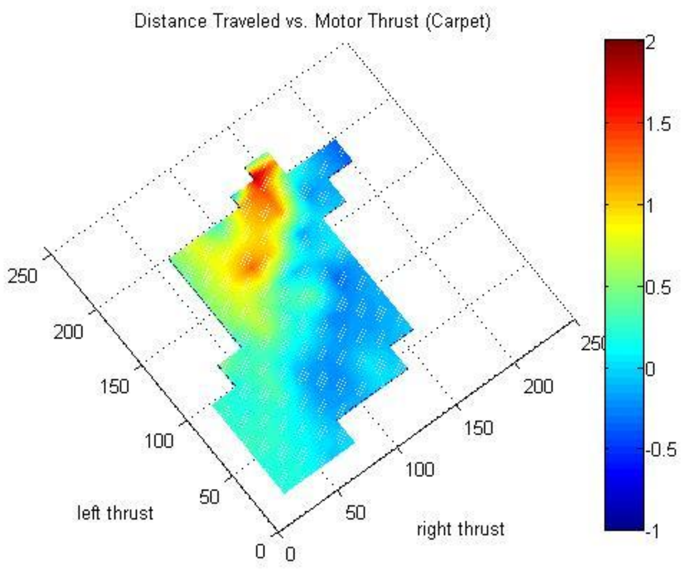


Fig. D8. Distance travelled vs motor thrusts on carpet surface.

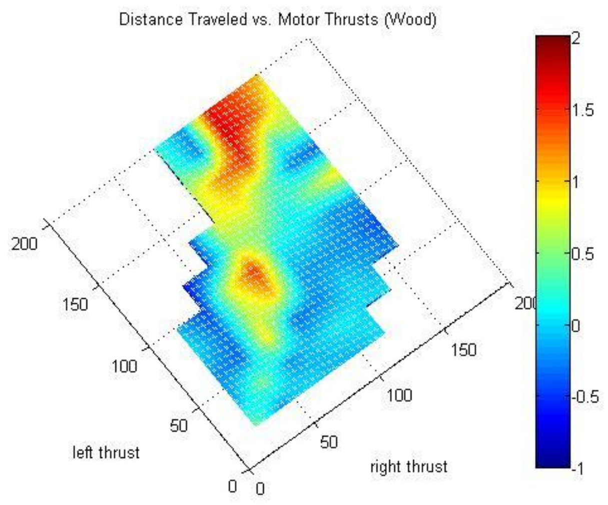


Fig. D9. Distance travelled vs motor thrusts on wood surface.

Based on the results above, there seems to be an interesting result in the relationship between forward distance travelled and motor thrust inputs on each different surfaces. The robot responded to the thrust inputs similarly on both surfaces, although with a slight translation in the policy space.

Likelihood ratio policy gradient RL method gave us some interesting results. Given an initial and final x, y positions, the speed, accuracy and reward function (harmonic mean, R) as calculated in the methodology section, it turns out that the likelihood-ratio method returned results comparable to those found through our hand-tuning experiments, with speeds of 0.1m/s and an accuracy of 99.5%.

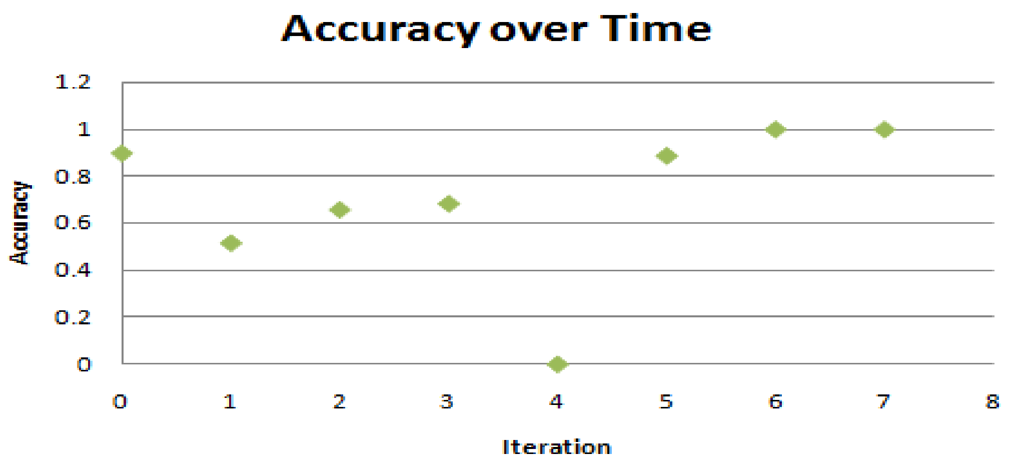


Fig. D10. Accuracy over time.

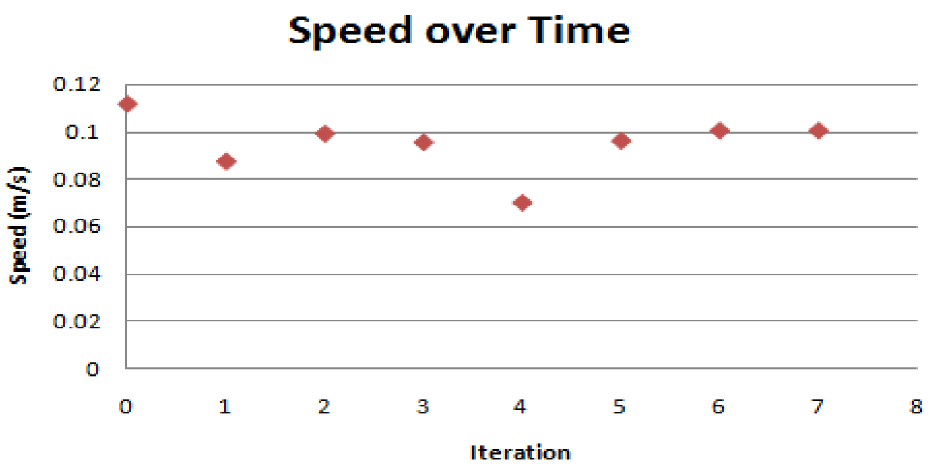


Fig. D11. Speed over time.

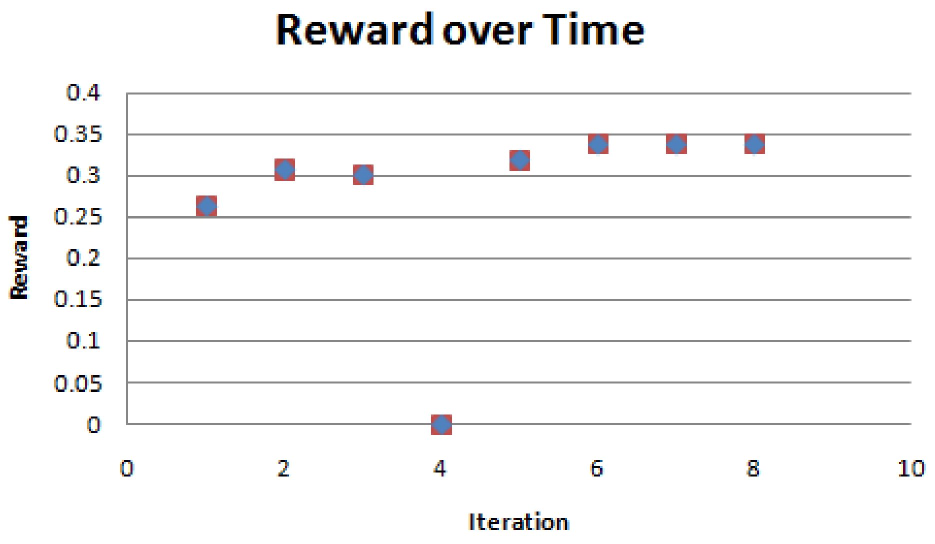


Fig. D12. Reward over time.

Our project confirms previous work, in which the walking efficiency of the robot is affected by many internal and external factors, such as mechanical inconsistencies, terrains with different textures and frictions, and many others.

Our project can be applied in the broader context, in the sense that hand tuning policies can be frustrating and tedious, especially when the number of robots increases. Thus, it is advised to implement reinforcement learning algorithms to not only enable the robots to autonomously walk across different terrains efficiently, but also reduce human intervention and time spent on tuning the robots. This will definitely help in increasing search and rescue operation efficiency and effectiveness.

Conclusion / Impact

In conclusion, running the OctoRoACH with (i) open loop system is tedious, time-consuming, generates random trajectories, and may not work well over time, across terrains and robots, and (ii) closed loop gyro feedback system is often better than that of the open loop, but still tedious and time-consuming when tuning PID controller and may not work well over time, across terrains and robots.

Thus, adaptive gait control is necessary to allow a legged robot to adapt itself to different terrains and walk efficiently. Specifically, we believe that RL methods may be effective to achieve these goals. Several RL methods, specifically the policy gradient methods, have been studied to allow control policy parameters to be automatically tuned. The best reinforcement learning algorithm that we have implemented on OctoRoACH thus far is the likelihood ratio policy gradient method. This method is dependent on the initial motor thrusts and provides fast convergence rate to a local optimum and eventually global optimum to enable robots to walk faster and straighter on different terrains. With RL implementation, robots can potentially assist human rescuers to perform reconnaissance, or search and rescue operations efficiently and ultimately save more lives during disaster situations.

In future, policy search methods can be developed based on characterized behaviours: (i) behaviour of robots on a variety of terrains and (ii) behaviour of several robots. Robots may be wirelessly connected to one another, provide real-time audiovisual feedback to the base station and locomote in swarms to expand the search and rescue network effectively and efficiently. Electronic and mechanical body parts of robots can be standardized with higher quality materials, so that more time can be spent implementing useful advanced machine learning algorithms instead of fixing and tuning the actual robot.

References

1. Aaron M. Hoover, Samuel Burden, Xiao-Yu Fu, S. Shankar Sastry, R. S. Fearing. "Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot", IEEE BioRob, Tokyo (September, 2010)
2. J. Zico Kolter, Mike P. Rodgers, Andrew Y. Ng. "A Control Architecture for Quadruped Locomotion Over Rough Terrain", In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2008)
3. J. Zico Kolter, Andrew Y. Ng. "Learning Omnidirectional Path Following Using Dimensionality Reduction", In Proceedings of Robotics: Science and Systems (RSS) (2007)
4. Manish Saggar, Thomas D'Silva. Nate Kohl and Peter Stone. "Autonomous Learning of Stable Quadruped Locomotion", in World (2006-2007)
5. Nate Kohl and Peter Stone. "Machine Learning for Fast Quadrupedal Locomotion", In Proceedings of AAAI, (2004)
6. Nate Kohl, Peter Stone. "Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion", IEEE International Conference on Robotics and Automation 2004 Proceedings ICRA (April 2004)
7. Andrew Ng, Michael Jordan. "PEGASUS: A policy search method for large MDPs and POMDPs", In Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference (2000)
8. Richard S. Sutton, Andrew G. Barto. "Reinforcement Learning, Sutton and Barto", A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England (2005)
9. Jie Tang, Pieter Abbeel. "On a Connection between Importance Sampling and the Likelihood Ratio Policy Gradient", In Advances in Neural Information Processing Systems (2010)
10. A.O. Pullin, N.J. Kohut, D. Zarrouk, R. S. Fearing. "Dynamic turning of 13 cm robot comparing tail and differential drive", to appear IEEE Int. Conf. Robotics and Automation (May 2012)
11. http://www.scholarpedia.org/article/Policy_gradient_methods#Likelihood_Ratio_Methods_and_REINFORCE
12. Hsiao, K., Kaelbling, L., Lozano-Perez, T. "Grasping POMDPs", IEEE Conference on Robotics and Automation (ICRA), (2007)
13. Malcolm J A Strens, Andrew W Moore. "Direct Policy Search using Paired Statistical Tests", In Proceedings of the 18th International Conference on Machine Learning (2001)
14. M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21., http://cvxr.com/cvx, April 2011.
15. M. Grant and S. Boyd. "Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control", V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, Lecture Notes in Control and Information Sciences, Springer, http://stanford.edu/~boyd/graph_dep.html (2008)
16. A. W.Coburn, R.J.S.Spence & A.Pomonis, "Factors determining human casualty levels in earthquakes: Mortality prediction in building collapse", Earthquake Engineering, Tenth World Conference, Balkema, Rotterdam (1992)
17. Gerald Tesauro, Gregory R. Galperin. "On-line Policy Improvement using Monte-Carlo Search", In Advances in Neural Information Processing Systems, 1068-1074, Cambridge, MA, (1996).