Learning Image-Conditioned Dynamics Models for Control of Under-actuated Legged Millirobots

Anusha Nagabandi

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-43
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-43.html

May 10, 2018

Acknowledgement

This report includes contributions from graduate student Gregory Kahn, undergraduate students Guangzhao
Yang and Thomas Asmar, and professors Sergey Levine and Ronald S. Fearing. We would like to thank
Carlos Casarez for extensive in-lab training of VelociRoACH construction, constant mechanical support,
general expertise with these systems, and many insightful discussions. This work is supported by the National
Science Foundation under the National Robotics Initiative, Award CMMI-1427096.

Learning Image-Conditioned Dynamics Models for Control of Under-actuated Legged Millirobots

by Anusha Nagabandi

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of Master of Science, Plan II.

Committee:

Professor Ronald S. Fearing
Research Co-Advisor

(Date)

Professor Sergey Levine
Research Co-Advisor

(Date)

Abstract

Millirobots are a promising robotic platform for many applications due to their small size and low manufacturing costs. Legged millirobots, in particular, can provide increased mobility in complex environments and improved scaling of obstacles. However, controlling these small, highly dynamic, and underactuated legged systems is difficult. Hand-engineered controllers can sometimes control these legged millirobots, but they have difficulties with dynamic maneuvers and complex terrains. We present an approach for controlling a real-world legged millirobot that is based on learned neural network models. Using less than 17 minutes of data, our method can learn a predictive model of the robot's dynamics that can enable effective gaits to be synthesized on the fly for following user-specified waypoints on a given terrain. Furthermore, by leveraging expressive, high-capacity neural network models, our approach allows for these predictions to be directly conditioned on camera images, endowing the robot with the ability to predict how different terrains might affect its dynamics. This enables sample-efficient and effective learning for locomotion of a dynamic legged millirobot on various terrains, including gravel, turf, carpet, and styrofoam. Experiment videos can be found at `https://sites.google.com/view/imageconddyn` [1]

[1]This report includes contributions from graduate student Gregory Kahn, undergraduate students Guangzhao Yang and Thomas Asmar, and professors Sergey Levine and Ronald S. Fearing.

Contents

Chapter 1

Introduction

Legged millirobots are an effective platform for applications, such as exploration, mapping, and search and rescue, because their small size and mobility allows them to navigate through complex, confined, and hard-to-reach environments that are often inaccessible to aerial vehicles and untraversable by wheeled robots. Millirobots also provide additional benefits in the form of low power consumption and low manufacturing costs, which enables scaling them to large teams that can accomplish more complex tasks. This superior mobility, accessibility, and scalability makes legged millirobots some of the most mission-capable small robots available. However, the same properties that enable these systems to traverse complex environments are precisely what make them difficult to control.

Modeling the hybrid dynamics of under-actuated legged millirobots from first principles is exceedingly difficult due to complicated ground contact physics that arise while moving dynamically on complex terrains. Furthermore, cheap and rapid manufacturing techniques cause each of these robots to exhibit varying dynamics. Due to these modeling challenges, many locomotion strategies for such systems are hand-engineered and heuristic. These manually designed controllers impose simplifying assumptions, which not only constrain the overall capabilities of these platforms, but also impose a heavy burden on the engineer. Additionally, and perhaps most importantly, they preclude the opportunity for adapting and improving over time.

In this work, we explore how learning can be used to automatically acquire locomotion strategies in diverse environments for small, low-cost, and highly dynamic legged robots. Choosing an appropriate learning algorithm requires consideration of a number of factors. First, the learned model needs to be expressive enough to cope with the highly dynamic and nonlinear nature of legged millirobots, as well as with high-dimensional sensory observations such as images. Second, the algorithm must allow the robot to learn quickly from modest amounts of data, so as to make it a practical algorithm for real-world application. Third, the learned general-purpose models must be able to be deployed on a wide range of navigational tasks in a diverse set of environments, with minimal human supervision.

The primary contribution of our work is an approach for controlling dynamic legged millirobots that learns an expressive and high-dimensional image-conditioned neural network dynamics model, which is then

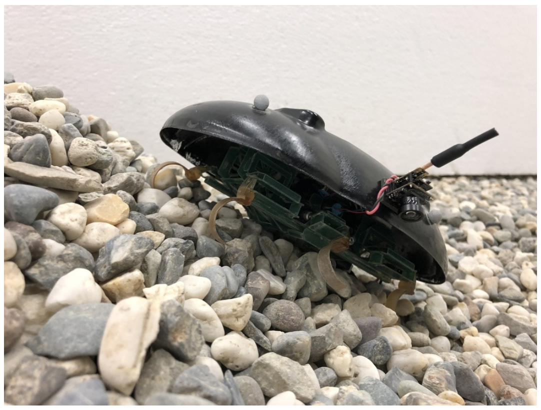


Figure 1.1: VelociRoACH: the small, mobile, highly dynamic, and bio-inspired hexapedal millirobot used in this work, shown with a camera mounted for terrain imaging.

combined with a model predictive controller (MPC) to follow arbitrary paths. Our sample efficient learning-based approach uses less than 17 minutes of real-world data to learn to follow desired paths in a desired environment, and we empirically show that it outperforms a conventional differential drive control strategy for highly dynamic maneuvers. Our method also enables adaptation to diverse terrains by conditioning its dynamics predictions on its own observed images, allowing it to predict how terrain features such as gravel or turf will alter the system's response. To the best of our knowledge, we believe this work is the first to leverage and build upon recent advances in learning to achieve a high-performing and sample efficient approach for controlling dynamic legged millirobots.

Chapter 2

Related Work

Controlling Legged Millirobots: Extensive prior work on controlling legged robots includes larger legged robots such as *Anymal* [1], *ASIMO* [2], and *Big Dog* [3]. These systems can achieve successful locomotion, but they have multiple degrees of freedom per leg and a relatively slow stride frequency that allows for more sophisticated control strategies of planned foot placement [4, 5, 6, 7]. Other prior work includes systems such as RHex [8], where each leg has an independent actuator and can thus execute stable alternating tripod gaits to achieve desired motion. Unlike these systems, however, we are interested in dynamic legged millirobots that are underactuated; these descriptors imply that we cannot move each leg independently, that we have neither the ability nor time to plan specific foot placement, and that we cannot strive for static or quasi-static gaits where stability and well-behaved dynamics can be expected. This realm of steering methods for dynamic running of underactuated legged millirobots includes various methods [9, 10], such as actively changing leg kinematics [11, 12], modulating leg impedance [13], and executing roll oscillation modulated turning [14]. However, these approaches achieve open-loop turning gaits, while we desire a closed-loop approach to precise path execution. Other traditional methods for both control and modeling of legged systems make simplifying assumptions, such as approximating a system as a spring loaded inverted pendulum (SLIP) model [15, 16] or approximating a system's behavior with a differential drive control strategy. Although these approaches do succeed in certain regimes [17], they fail when high speeds or irregular environments lead to more complicated dynamics. In contrast, our neural network learning-based approach can cope with complex dynamics, while also incorporating high-dimensional environmental information in the form of images.

Gait Optimization: Instead of building on simplifying model assumptions to design controllers, prior work has also explored various methods of automatic gait optimization [18, 19]. These methods include stochastic gradient descent [20], genetic algorithms [21], and Bayesian optimization [22, 23, 24] to reduce the time-consuming design process of manually finding robust parameters. For instance, [20] optimized a control policy for bipedal walking online in less than 20 minutes on a simplified system with 6 joints, and [19] learned model-free sensory feedback controllers to supplement specified open-loop gaits. While these methods are sample efficient and can be applied to real systems, they have not yet been shown to work for high dimensional systems or more complex systems, such as fast robots operating in highly dynamic regimes on irregular surfaces with challenging contact dynamics.

Model-free Policy Learning: Rather than optimizing gaits, prior work in model-free reinforcement learning algorithms has demonstrated the ability to instead learn these behaviors from scratch. Work in this area, including Q-learning [25, 26], actor-critic methods [27, 28], and policy gradients [29], has learned complex skills in high-dimensional state spaces, including skills for simulated robotic locomotion tasks. However, the high sample complexity of such purely model-free algorithms makes them difficult to use for learning in the real world, where sample collection is limited by time and other physical constraints. To our knowledge, no prior method has attempted model-free deep reinforcement learning of locomotion skills in the real-world, but Gu et al. [30] learn reaching skills with a robotic arm using several hours of experience. Unlike these approaches, our model-based learning method uses only minutes of experience to achieve generalizable real-world locomotion skills that were not explicitly seen during training, and it further exemplifies the benefits in sample complexity that arise from incorporating models with learning-based approaches.

Model Learning: Although the sample efficiency of model-based learning is appealing, and although data-driven approaches can eliminate the need to impose restrictive assumptions or approximations, the challenge lies in the difficulty of learning a good model. Relatively simple function approximators such as time-varying linear models have been used to model dynamics of systems [31, 32], including our VelociRoACH [33] platform. However, these models have not yet been shown to possess enough representational power (i.e., accuracy) to generalize to complex locomotion tasks. Prior work has also investigated learning probabilistic dynamics models [34, 35], including Gaussian process models for simulated legged robots [36]. However, to the best of our knowledge, no prior work has learned Gaussian process models for real-time control of dynamic real-world legged robots from raw data. Also, while these approaches can be sample efficient, it is intractable to scale them to higher dimensions, as needed especially when incorporating rich sensory inputs such as image observations. In contrast, our method employs expressive neural network dynamics models, which easily scale to high dimensional inputs. Other modeling approaches have leveraged smaller neural networks for dynamics modeling, but they impose strict and potentially restrictive structure to their formulation, such as designing separate modules to represent the various segments of a stride [37], approximating actuators as muscles and tuning these parameters [38], or calculating equations of motion and learning error terms on top of these specific models [39]. Instead, we demonstrate a sample efficient, expressive, and high-dimensional neural network dynamics model that is free to learn without the imposition of an approximated hand-specified structure.

Environment Adaptation: The dynamics of a robot depend not only on its own configuration, but also on its environment. Prior methods generally categorize the problem of adapting to diverse terrains into two stages: first, the terrain is recognized by a classifier trained with human-specified labels (or, less often, using unsupervised learning methods [40]), and second, the gait is adapted to the terrain. This general approach has been used for autonomous vehicles [41, 40], larger legged robots [5, 6, 7, 38, 42], and for legged millirobots [43, 44]. In contrast, our method does not require any human labels at run time, and it adapts to terrains based entirely on autonomous exploration: the dynamics model is simply conditioned on image observations of the terrain, and it automatically learns to recognize the visual cues of terrain features that affect the robot's dynamics.

This work: While our prior work evaluated model-based reinforcement learning with neural network models [45], to our knowledge, the present work is the first to extend these model-based learning techniques to real-world robotic locomotion on various terrains. Furthermore, we present a novel extension of this approach that conditions the dynamics predictions on image observations and allows for adaptation to various terrain types.

Chapter 3

Model-Based Learning Method for Locomotion Control

In this work, we propose an automated method of acquiring locomotion strategies for small, low-cost, dynamic legged millirobots. In this section, we describe a method for learning a neural network dynamics model [45] (Sec. 3.1), using the model as part of a model predictive controller (Sec. 3.2), and extending the model into an image-conditioned model using features from a pre-trained convolutional neural network. Fig. 3.1 provides an overview of our method.

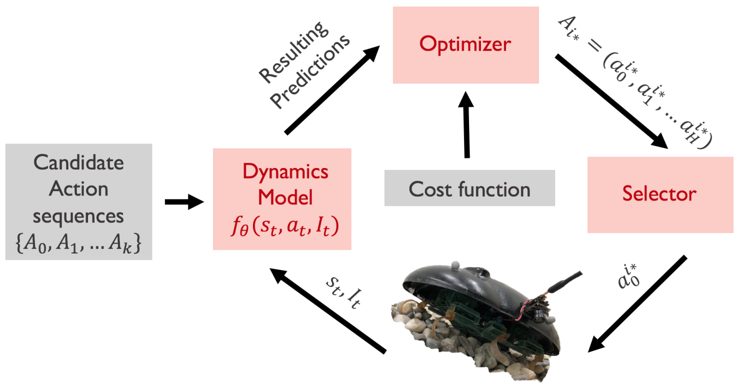

Candidate Action sequences $\{A_0, A_1, \dots A_k\}$
Dynamics Model $f_\theta(s_t, a_t, I_t)$
Resulting Predictions
Optimizer
Cost function
$A_{i*} = (a_0^{i*}, a_1^{i*}, \dots a_H^{i*})$
Selector
a_0^{i*}
s_t, I_t

Figure 3.1: Our image-conditioned model-based learning method for locomotion control: A closed-loop MPC controller uses predictions from the learned dynamics model to perform action selection.

3.1 Learning System Dynamics

We require a parameterization of the dynamics model that can cope with high-dimensional state and action spaces, and the complex dynamics of legged millirobots. We therefore represent the dynamics function $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$ as a multilayer neural network, parameterized by θ. This function outputs the predicted change in state that occurs as a result of executing action $\mathbf{a}_t$ from state $\mathbf{s}_t$, over the time step duration of Δt. Thus, the predicted next state is given by: $\hat{\mathbf{s}}_{t+1} = \mathbf{s}_t + \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$. While choosing too small of a Δt leads to too small of a state difference to allow meaningful learning, increasing the Δt too much can also make the learning process more difficult because it increases the complexity of the underlying continuous-time dynamics. Although we do not perform a structured study of various Δt values for our system, we provide this insight as something for consideration when implementing this method on other systems.

We define the state $\mathbf{s}_t$ of the VelociRoACH to be $[x,\ y,\ z,\ v_x,\ v_y,\ v_z,\ \cos(\phi_r),\ \sin(\phi_r),\ \cos(\phi_p),$ $\sin(\phi_p),\ \cos(\phi_y),\ \sin(\phi_y),\ \omega_x,\ \omega_y,\ \omega_z,\ \cos(aL),\ \sin(aL),\ \cos(aR),\ \sin(aR),\ v_{aL},\ v_{aR},\ \mathrm{b}emf_L,\ \mathrm{b}emf_R,$ $V_{bat}]^T$. The center of mass positions (x, y, z) and the Euler angles to describe the center of mass pose (ϕ_r, ϕ_p, ϕ_y) come from the OptiTrack motion capture system. The angular velocities $(\omega_x, \omega_y, \omega_z)$ come from the gyroscope onboard the IMU, and the motor crank positions (aL, aR) come from the magnetic rotary encoders, which give a notion of leg position. We include $(\mathrm{b}emf_L, \mathrm{b}emf_R)$ because back-EMF provides a notion of motor torque/velocity, and (V_{bat}) because the voltage of the battery affects the VelociRoACH's performance. Note that the state includes *sin* and *cos* of angular values, which is common practice and allows the neural network to avoid wrapping issues.

We define the action representation of the VelociRoACH to represent desired velocity setpoints for the rotation of the legs, and we achieve these setpoints using a lower-level PID controller onboard the system. We collect training data by placing the robot in arbitrary start states and executing random actions at each time step. We record each resulting trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, \cdots, \mathbf{s}_{T-2}, \mathbf{a}_{T-2}, \mathbf{s}_{T-1})$ of length T. We slice the trajectories $\{\tau\}$ into training data inputs $(\mathbf{s}_t, \mathbf{a}_t)$ and corresponding output labels $(\mathbf{s}_{t+1} - \mathbf{s}_t)$. We train the dynamics model $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$ on data from the training dataset $\mathcal{D}$ by minimizing the error

$$\mathcal{E}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}} \frac{1}{2} \|(\mathbf{s}_{t+1} - \mathbf{s}_t) - \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)\|_2^2 \tag{3.1}$$

using stochastic gradient descent. Prior to training, we preprocess the training data by normalizing it to be mean 0 and standard deviation 1, which ensures equal weighting of different state elements, regardless of their magnitudes.

3.2 Model-Based Control Using Learned Dynamics

We formulate a model-based controller which uses the learned model $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$ together with a cost function $c(\mathbf{s}_t, \mathbf{a}_t)$ that encodes some task. Many methods could be used to perform this action selection, and we use a random-sampling shooting method [46]. At each time step t, we randomly generate K candidate action sequences of H actions each, use the learned dynamics model to predict the resulting states, and then use the cost function to select the action sequence with the lowest cost. The cost function that we use for path following is as follows:

$$c(\mathbf{s}_t, \mathbf{a}_t) = f_p * p + f_h * h + f_f * f, \tag{3.2}$$

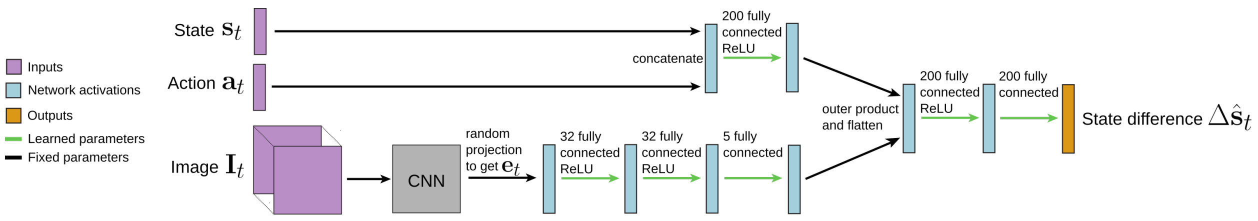


Figure 3.2: Our image-conditioned neural network dynamics model. The model takes as input the current state $\mathbf{s}_t$, action $\mathbf{a}_t$, and image $\mathbf{I}_t$. The image is passed through the convolutional layers of AlexNet [47] pre-trained on ImageNet [48], which is then flattened and projected into a lower dimension through multiplication with a random fixed matrix to obtain $\mathbf{e}_t$. The image and concatenated state-action vectors are passed through fully connected layers, fused via an outer product, flattened, and passed through more fully connected layers to obtain the predicted state difference $\Delta\hat{\mathbf{s}}_t$.

where the parameter f_p penalizes perpendicular distance p away from the desired path, parameter f_f encourages forward progress f along the path, and parameter f_h maintains the heading h of the system toward the desired direction. Rather than executing the entire sequence of selected optimal actions, we use model predictive control (MPC) to execute only the first action $\mathbf{a}_t$, and we then replan at the next time step, given updated state information.

3.3 Image-Conditioned Dynamics Model

As currently described, our model-based learning approach can successfully follow arbitrary paths when trained and tested on a single terrain. However, in order to traverse complex and varied terrains, it is necessary to adjust the dynamics to the current terrain conditions. One approach to succeeding in multiple environments would be to train a separate dynamics model for each terrain. However, in addition to requiring many separate models, this would lead to models that would likely generalize poorly. Furthermore, this approach would require a person to label the training data, as well as each run at test-time, with which terrain the robot is in. All of these aspects are undesirable for an autonomous learning system.

Instead, we propose a simple and highly effective method for incorporating terrain information, using only observations from a monocular color camera mounted on the robotic platform. We formulate an image-conditioned dynamics model $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t, \mathbf{I}_t)$ that takes as input not only the current robot state $\mathbf{s}_t$ and action $\mathbf{a}_t$, but also the current image observation $\mathbf{I}_t$. The model (Fig. 3.2) passes image $\mathbf{I}_t$ through the first eight layers of AlexNet [47]. The resulting activations are flattened into a vector, and this vector is then multiplied by a fixed random matrix in order to produce a lower dimensional feature vector $\mathbf{e}_t$. The concatenated state-action vector $[\mathbf{s}_t; \mathbf{a}_t]$ is passed through a hidden layer and combined with $\mathbf{e}_t$ through an outer product. As opposed to a straightforward concatenation of $[\mathbf{s}_t; \mathbf{a}_t; \mathbf{e}_t]$, this outer product allows for higher-order integration of terrain information terms with the state and action information terms. This combined layer is then passed through another hidden layer and output layer to produce a prediction of state difference $\Delta\hat{\mathbf{s}}_t$.

Training the entire image-conditioned neural network dynamics model with only minutes of data—

corresponding to tens of thousands of datapoints—and in only a few environments would result in catastrophic overfitting. Thus, to perform feature extraction on our images, we use the AlexNet [47] layer weights optimized from training on the task of image classification on the ImageNet [48] dataset, which contains 15 million diverse images. Although gathering and labelling this large image dataset was a significant effort, we note that such image datasets are ubiquitous and their learned features have been shown to transfer well to other tasks [49]. By using these pre-trained and transferable features, our image-conditioned dynamics model is sample-efficient and can automatically adapt to different terrains without any manual labelling of terrain information.

We show in our experiments that this image-conditioned dynamics model outperforms a naïvely trained dynamics model that is trained simply on an aggregation of all the data. Furthermore, the performance of our image-conditioned dynamics model is comparable, on each terrain, to individual dynamics models that are specifically trained (and tested) on that terrain.

Chapter 4

Results

The goal of our experimental evaluation is to study how well our model-based learning algorithm can control a real-world VelociRoACH to follow user-defined paths on various surfaces.

4.1 VelociRoACH Platform

The VelociRoACH is a minimally actuated, small, legged, and highly dynamic palm-sized robotic platform [50]. Compared to wheeled/tracked robots of similar size (Fig. 4.1), this legged system is able to successfully navigate over more complex terrains.

The VelociRoACH is constructed through a rapid manufacturing process known as smart composite microstructure (SCM) process [51]. This process allows for the creation of lightweight linkages, enabling the

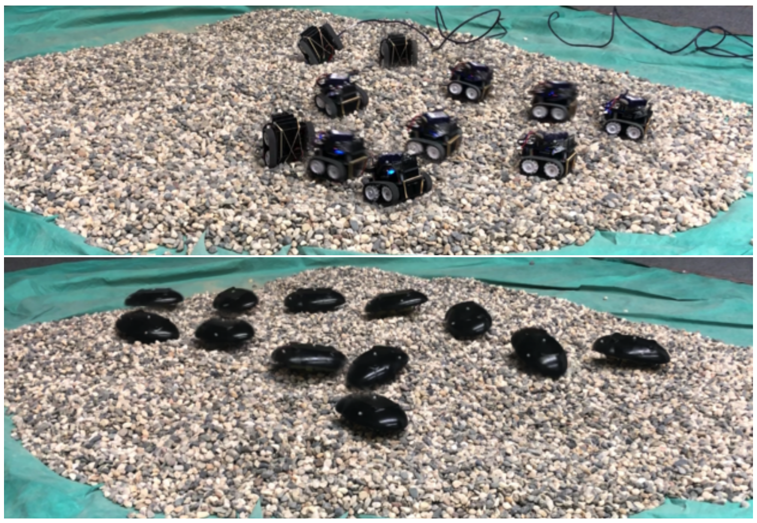


Figure 4.1: Over 15 teleoperated trials performed on rough terrain, a legged robot succeeded in navigating through the terrain 90% of the time, whereas a tracked robot of comparable size succeeded only 30% of the time.

rapid realization of fully functional prototypes of folded flexure-based mobile millirobots. The VelociRoACH's robot chassis can be constructed for just \$2, and this rigid structural core houses the battery, two motors, transmission, microcontroller, and all sensors. The core also provides mechanical grounding points for the kinematic linkages, which couple each of the two motors to three legs in order to reduce the number of required actuators.

The VelociRoACH carries an ImageProc embedded circuit board[1], which includes a 40 MHz Microchip dsPIC33F microprocessor, a six axis inertial measurement unit (IMU), an 802.15.4 wireless radio (XBee), and motor control circuitry. We added a 14-bit magnetic rotary encoders to the motors on each side of the robot to monitor absolute position. Additional sensory information includes battery voltage and back-EMF signals from the motors.

The onboard microcontroller runs a low-level 1 kHz control loop and processes communication signals from the XBee. Due to computational limits of the onboard microcontroller, we stream data from the robot to a laptop for calculating controller commands, and then stream these commands back to the onboard microcontroller for execution. To bypass the problem of using only on-board sensors for state estimation, we also use an OptiTrack motion capture system to stream robot pose information during experiments. The motion capture system does not provide any information about the environment terrain, so we also mounted a 3.4 gram monocular color camera onto the VelociRoACH, which communicates directly with the laptop via a radio frequency receiver and USB video converter.

4.2 Details of Our Approach

The learned dynamics function $\hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t, \mathbf{I}_t)$ is the neural network depicted in Fig. 3.2. For all experiments and results reported below, we use only 17 minutes (10,000 datapoints) of data from each terrain to train the dynamics model: This consists of 200 rollouts, each containing 50 data points that are collected at 10 Hz. We train each dynamics model for 50 epochs, using the Adam optimizer [52] with learning rate 0.001 and batchsize 1000.

Relevant parameters for our model-based controller are the number of candidate action sequences sampled at each time step $N = 500$, the amount of time represented by one time step $\Delta t = 0.1$ sec, the horizon $H = 4$, and parameters $f_p = 50$, $f_f = 10$, and $f_h = 5$ for the perpendicular, forward, and heading components of the cost function from Eqn. 3.2. To simplify training and testing of the image-conditioned dynamics model, the image at the start of the rollout was used for all timesteps. The process of using the neural network dynamics model and the cost function to select the best candidate action sequence at each time step can be done in real-time, even on a laptop with no GPU, and even taking bi-directional communication delays into account.

Note that the training data is gathered entirely using random trajectories, and therefore, the paths executed by our controller at run-time differ substantially from the training data. This illustrates that our approach can be trained with off-policy data, and that the model exhibits considerable generalization.

[1]`https://github.com/biomimetics/imageproc_pcb`

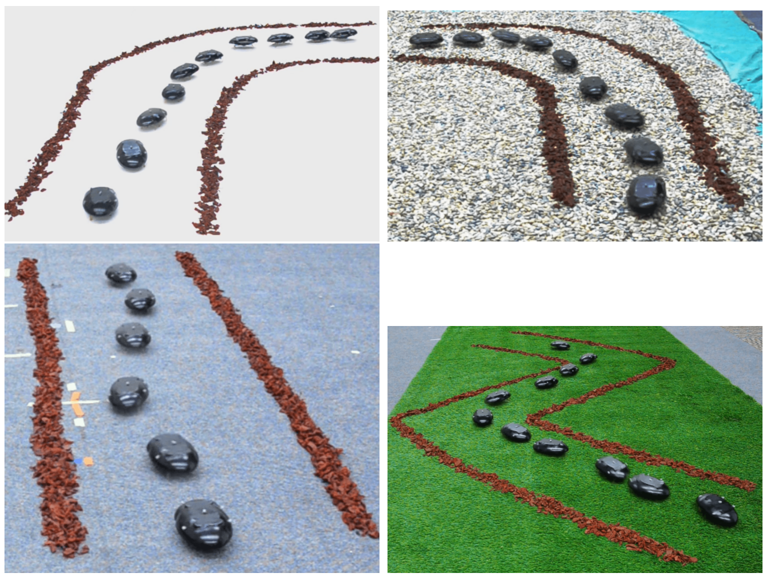


Figure 4.2: Execution of our model-based learning method, using an image-conditioned dynamics model, on various desired paths on the four terrains (styrofoam, gravel, carpet, and turf) we consider. Note that the path boundaries are outlined for visualization purposes only, and were not present during the experiments.

Furthermore, although the model is trained only once, we use it to accomplish a variety of tasks at run-time by simply changing the desired path in our cost function. This eliminates the need for task-specific training, which further improves overall sample efficiency. We show in Fig. 4.2 some images of the VelociRoACH using our model-based learning method to execute different paths on various surfaces.

4.3 Comparing to Differential Drive

To provide a comparison of our model-based learning algorithm's performance, we compare to a differential drive controller, which is a common steering method used for robots with wheel or leg-like mechanisms on both sides. A differential drive control strategy controls the system's heading by specifying the left and right leg velocities based on the system's perpendicular distance to the desired path: Moving the right wheels would turn the robot to the left, and moving the left wheels would turn the robot to the right.

In comparing our method to the differential drive controller, we tuned the differential drive controller hyperparameters in the same single environment that the model-based controller hyperparameters were tuned in. Also, all cost numbers reported below are calculated on the same cost function (Eqn. 3.2) that indicates how well the executed path aligns with the desired path, and each reported number represents an average over 10 runs.

Fig. 4.3 illustrates, on different paths executed on carpet, that our model-based learning method and the differential drive control strategy are comparable at low speeds. However, our model-based approach outperforms the differential drive strategy at higher speeds. The performance of differential drive deteriorates

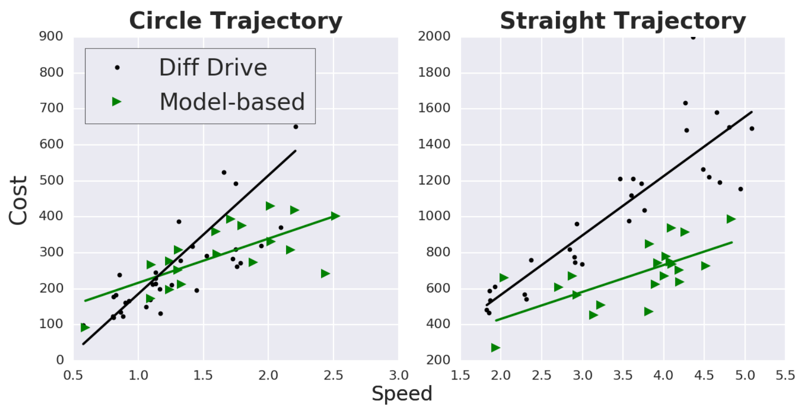


Figure 4.3: An analysis of cost incurred during trajectory following, as a function of the speed of the robot, shows that our model-based learning method is comparable to a differential drive control strategy at low speeds, but outperforms differential drive at high speeds. Note that each cost shown here is the sum of costs accumulated over 100 timesteps for each trajectory.

as leg speeds increase, because traction decreases and causes the legs to have less control over heading. Also, at high speeds, the dynamics of the legged robot can produce significant roll oscillations, depending on the leg phasing [14]. Therefore, based on the timing of left and right foot contacts, the system can produce turns inconsistent with a differential drive control strategy. Fig. 4.4 illustrates that for different paths across various surfaces, our model-based learning method outperforms the differential drive control strategy. Furthermore, we note that this difference in performance is most pronounced on surfaces with less traction, such as styrofoam and carpet.

4.4 Improving Performance with More Data

To investigate the effect of the quantity of training data, we trained three different dynamics models using different amounts of training data on carpet. We trained one with 50 rollouts (4 minutes), one with 200 rollouts (17 minutes), and one with 400 rollouts (32 minutes). Table 4.1 indicates that more training data can indeed improve task performance. This is an encouraging indication that improvement can occur over time, which is not the case for hand-engineered solutions.

	Straight	Left	Right
50 rollouts	14.4	16.6	29.4
200 rollouts	10.3	13.6	17.1
400 rollouts	10.8	11.3	11.5

Table 4.1: Cost incurred by the VelociRoACH during the task of trajectory following on carpet. Three models were trained, each with different amounts of training data, and they show performance improvements occurring over time (with more data). Here, one rollout corresponds to 50 timesteps or 5 seconds of data.

4.5 Learning Environmental Information

To verify whether our learned model encapsulates information about the environment, and to see whether or not the learned model itself has a large effect on controller performance, we conducted experiments on a carpet material and a slippery styrofoam material. Table 4.2 shows that the baseline differential drive controller performs relatively poorly on both surfaces. For the model-based approach, the model trained on the carpet works well on the carpet, and the model trained on the styrofoam works well on the styrofoam. The poor performance of either model on the other surface illustrates that our learned dynamics model does in fact encode some knowledge about the surface. Also, performance diminishes when the model is trained on data gathered from both terrains, which indicates that this naïve method for training a joint dynamics model is insufficient.

	Carpet	Styrofoam
Differential Drive	13.85	15.45
Model trained on carpet	5.69	18.62
Model trained on styrofoam	22.25	8.15
Model trained on both	7.52	15.76

Table 4.2: Costs incurred by the VelociRoACH while executing a straight line path. The model-based controller has the best performance when executed on the surface that it was trained on. Additionally, a model trained on carpet fails on styrofoam (and vice versa), indicating that the model incorporates some knowledge about the environment of operation. Furthermore, a model trained jointly on data from all surfaces does not result in good performance.

4.6 Image-Conditioned Dynamics Models

We have shown so far that when trained on data gathered from a single terrain, our model-based approach is superior to a standard differential drive approach, and that our approach improves with more data. However, although we saw that the robot's dynamics depend on the environment, we would like our approach to be able to control the VelociRoACH on a variety of terrains.

A standard approach would be to train a dynamics model using data from all terrains. However, as shown above in Table 4.2 as well as below in Fig. 4.4, a model that is naïvely trained on all data from multiple terrains and then tested on one of those terrains is significantly worse than a model that is trained solely on that particular terrain. The main reason that this naïve approach does not work well is that the dynamics themselves differ greatly with terrain, and a dynamics model that takes only the robot's current state and action as inputs receives a weak and indirect signal about the robot's environment.

To have a direct signal about the environment, our image-conditioned model takes an additional input: an image taken from an onboard camera, as described in Sec. 3.3. We compare our image-conditioned dynamics model to various alternate approaches, including (a) training a separate dynamics model on each terrain, (b) naïvely training one joint dynamics model on all training data, with no images or labels, and (c) training one joint dynamics model using data with explicit terrain labels **e** (Fig. 3.2) in the form of a one-hot vector (where the activation of a single vector element corresponds directly to operation in that terrain).

Fig. 4.4 compares the performance of our image-conditioned approach to that of these alternative approaches, on the task of path following for four different paths (straight, left, right, zigzag) on four different surfaces (styrofoam, carpet, gravel, turf). The naïve approach for training one joint dynamics model using an aggregation of all data performs worse than the other learning-based methods. The method of having a separate dynamics model for each terrain, as well as the method of training one joint dynamics model using one-hot vectors as terrain labels, both perform well on all terrains. However, both of these methods require human supervision to label the training data and to specify which terrain the robot is on at test time. In contrast, our image-conditioned approach performs just as well as the separate and one-hot models, but does not require any additional supervision beyond an onboard monocular camera. Finally, our image-conditioned approach also substantially outperforms the differential drive baseline on all terrains.

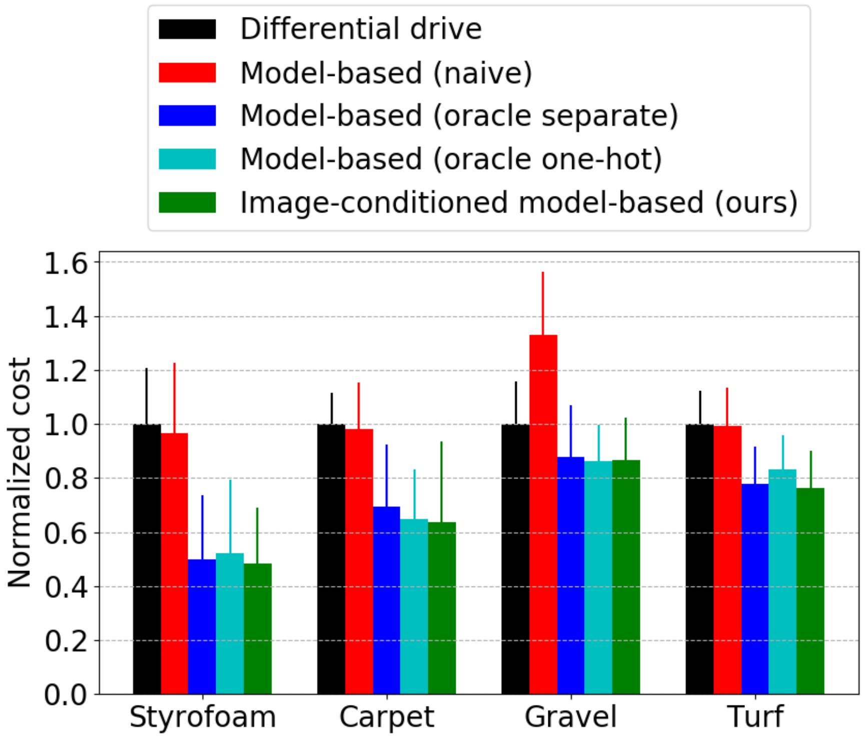


Figure 4.4: Comparison of our image-conditioned model-based approach to alternate methods. Each method was evaluated on four different terrains: styrofoam, carpet, gravel, and turf. On each terrain, four different paths (straight, left, right, and zigzag) were evaluated 10 times each. The methods that we compare to include: a hand-engineered differential drive controller, a joint dynamics model that is naïvely trained on all data from all terrains, an “oracle" approach that uses a separate dynamics model on each terrain, and another “oracle" approach where the joint dynamics model is trained using data containing an extra one-hot vector input indicating the terrain label of each data point. Our method outperforms the differential drive method and the naïve model-based controller, while performing similarly to the oracle baselines without needing any explicit labels.

Chapter 5

Discussion

We presented a sample-efficient model-based learning algorithm using image-conditioned neural network dynamics models that enables accurate locomotion of a low-cost, under-actuated, legged, and highly dynamic VelociRoACH robot in a variety of environments. Using only 17 minutes of real-world data for each terrain, our method outperformed a common differential drive control strategy, showed improvement with more data, and was able to use features from images in order to execute successful locomotion on various terrains.

One drawback of our method is the amount of computation involved at each step. We overcame the limitations of our embedded processor by streaming information to and from an external computer. However, performing all computations on-board would reduce delays, increase robustness to communication issues, and make this system more suitable for real-world tasks. One option could be to use the learned dynamics model to simulate rollouts of training data, which could then be used to train a policy (without more real-world data collection). However, this would require further algorithmic development, because the current dynamics model diverges after a few time steps, which precludes its direct applicability to traditional reinforcement learning algorithms that require longer rollouts for policy training.

Another direction for future work includes developing an algorithm for online adaptation of the learned model. This would improve performance because over time, the roach suffers from deterioration of the chassis, motor strength, and leg characteristics. Furthermore, online adaptation would allow the robot to succeed at test tasks further away from the training distribution, allowing for adaptation to both new tasks as well as to unexpected environmental perturbations. Additionally, removing the dependence on a motion capture system is compelling, particularly when aiming for real-world application.

Another interesting line of future work includes improving the MPC controller. Our current approach samples random actions at each time step and uses the predictions from the dynamics model to select the best action sequence. However, sampling based approaches are intractable for systems with high-dimensional action spaces over long time horizons. Furthermore, a more structured search of the action space could prevent rapidly changing actions, limit the search space to more meaningful options, and also enable the discovery of gaits through imposing cyclic or other intelligible constraints.

Chapter 6

Acknowledgements

This report includes contributions from graduate student Gregory Kahn, undergraduate students Guangzhao Yang and Thomas Asmar, and professors Sergey Levine and Ronald S. Fearing. We would like to thank Carlos Casarez for extensive in-lab training of VelociRoACH construction, constant mechanical support, general expertise with these systems, and many insightful discussions. This work is supported by the National Science Foundation under the National Robotics Initiative, Award CMMI-1427096.

Bibliography

[1] 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 *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2016, pp. 38–44.

[2] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent ASIMO: System overview and integration," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3. IEEE, 2002, pp. 2478–2483.

[3] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[4] K. Byl, "Metastable legged-robot locomotion," Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[5] J. Z. Kolter, P. Abbeel, and A. Y. Ng, "Hierarchical apprenticeship learning with application to quadruped locomotion," in *Advances in Neural Information Processing Systems*, 2008, pp. 769–776.

[6] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 2665–2670.

[7] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, J. A. Bagnell, C. G. Atkeson, and J. Kuffner, "Optimization and learning for rough terrain legged locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 175–191, 2011.

[8] R. Altendorfer, N. Moore, H. Komsuoglu, M. Buehler, H. Brown, D. McMordie, U. Saranli, R. Full, and D. E. Koditschek, "RHex: A biologically inspired hexapod runner," *Autonomous Robots*, vol. 11, no. 3, pp. 207–213, 2001.

[9] A. J. McClung III, *Techniques for dynamic maneuvering of hexapedal legged robots*, 2006, vol. 67, no. 11.

[10] D. Zarrouk, D. W. Haldane, and R. S. Fearing, "Dynamic legged locomotion for palm-size robots," in *SPIE Defense+ Security.* International Society for Optics and Photonics, 2015, pp. 94 671S–94 671S.

[11] J. E. Clark, J. G. Cham, S. A. Bailey, E. M. Froehlich, P. K. Nahata, R. J. Full, and M. R. Cutkosky, "Biomimetic design and fabrication of a hexapedal running robot," in *IEEE Int. Conf. on Robotics and Automation*, vol. 4, 2001, pp. 3643–3649.

[12] S. Kim, J. E. Clark, and M. R. Cutkosky, "iSprawl: Design and tuning for high-speed autonomous open-loop running," *The International Journal of Robotics Research*, vol. 25, no. 9, pp. 903–912, 2006.

[13] A. M. Hoover, S. Burden, X.-Y. Fu, S. S. Sastry, and R. S. Fearing, "Bio-inspired design and dynamic maneuverability of a minimally actuated six-legged robot," in *IEEE RAS and EMBS Int. Conf. on Biomedical Robotics and Biomechatronics*, 2010, pp. 869–876.

[14] D. W. Haldane and R. S. Fearing, "Roll oscillation modulated turning in dynamic millirobots," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 4569–4575.

[15] H. Komsuoglu, K. Sohn, R. J. Full, and D. E. Koditschek, "A physical model for dynamical arthropod running on level ground," *University of Pennsylvania, Departmental Papers (ESE)*, p. 466, 2008.

[16] H. Komsuoglu, A. Majumdar, Y. O. Aydin, and D. E. Koditschek, "Characterization of dynamic behaviors in a hexapod robot," in *Experimental Robotics.* Springer, 2014, pp. 667–684.

[17] R. Altendorfer, D. E. Koditschek, and P. Holmes, "Stability analysis of a clock-driven rigid-body SLIP model for RHex," *The International Journal of Robotics Research*, vol. 23, no. 10-11, pp. 1001–1012, 2004.

[18] X. Da, R. Hartley, and J. Grizzle, "Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field," in *ICRA*, 2017.

[19] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013.

[20] R. Tedrake, T. W. Zhang, and H. S. Seung, "Learning to walk in 20 minutes," in *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, vol. 95585. Yale University New Haven (CT), 2005, pp. 1939–1412.

[21] S. Chernova and M. Veloso, "An evolutionary approach to gait learning for four-legged robots," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, vol. 3, 2004, pp. 2562–2567.

[22] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "An experimental comparison of bayesian optimization for bipedal locomotion," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 1951–1958.

[23] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, "Automatic gait optimization with gaussian process regression." in *IJCAI*, vol. 7, 2007, pp. 944–949.

[24] M. Tesch, J. Schneider, and H. Choset, "Using response surfaces and expected improvement to optimize snake robot gait parameters," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2011, pp. 1069–1074.

[25] V. Mnih and et. al., "Human-level control through deep reinforcement learning," *Nature*, 2015.

[26] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Memorybased control of active perception and action in minecraft." ICML, 2016.

[27] T. P. Lillicrap and et. al., "Continuous control with deep reinforcement learning," *ICLR*, 2016.

[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016.

[29] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015.

[30] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *ICRA*, 2017.

[31] R. Lioutikov, A. Paraschos, J. Peters, and G. Neumann, "Sample-based information-theoretic stochastic optimal control," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[32] M. C. Yip and D. B. Camarillo, "Model-less feedback control of continuum manipulators in constrained environments," in *IEEE Transactions on Robotics*, 2014.

[33] A. D. Buchan, D. W. Haldane, and R. S. Fearing, "Automatic identification of dynamic piecewise affine models for a running robot," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2013, pp. 5600–5607.

[34] M. Deisenroth and C. Rasmussen, "A model-based and data-efficient approach to policy search," in *ICML*, 2011.

[35] J. Ko and D. Fox, "GP-BayesFilters: Bayesian filtering using gaussian process prediction and observation models," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2008.

[36] M. P. Deisenroth, R. Calandra, A. Seyfarth, and J. Peters, "Toward fast policy search for learning legged locomotion," in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2012, pp. 1787–1792.

[37] H. Cruse, T. Kindermann, M. Schumm, J. Dean, and J. Schmitz, "Walknet—a biologically inspired network to control six-legged walking," in *Neural Networks*, 1998.

[38] X. Xiong, F. Worgotter, and P. Manoonpong, "Neuromechanical control for hexapedal robot walking on challenging surfaces and surface classification," in *RAS*, 2014.

[39] R. Grandia, D. Pardo, and J. Buchli, "Contact invariant model learning for legged robot locomotion," in *RAL*, 2018.

[40] B. Leffler, “Perception-based generalization in model-based reinforcement learning,” Ph.D. dissertation, Rutgers, 2009.

[41] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, “Stanley: The robot that won the darpa grand challenge,” *Journal of Field Robotics*, 2006.

[42] M. A. Hoepflinger, C. D. Remy, M. Hutter, L. Spinello, and R. Siegwart, “Haptic terrain classification for legged robots,” in *ICRA*, 2010.

[43] X. A. Wu, T. M. Huh, R. Mukherjee, and M. Cutkosky, “Integrated ground reaction force sensing and terrain classification for small legged robots,” *RAL*, 2016.

[44] F. L. G. Bermudez, R. C. Julian, D. W. Haldane, P. Abbeel, and R. S. Fearing, “Performance analysis and terrain classification for a legged robot over rough terrain,” in *IROS*, 2012.

[45] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” *arXiv preprint arXiv:1708.02596*, 2017.

[46] A. Rao, “A survey of numerical methods for optimal control,” in *Advances in the Astronautical Sciences*, 2009.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *CVPR*, 2009.

[49] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “Cnn features off-the-shelf: an astounding baseline for recognition,” in *CVPR Workshops (CVPRW)*, 2014.

[50] D. W. Haldane, K. C. Peterson, F. L. G. Bermudez, and R. S. Fearing, “Animal-inspired design and aerodynamic stabilization of a hexapedal millirobot,” in *IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 3279–3286.

[51] A. M. Hoover and R. S. Fearing, “Fast scale prototyping for folded millirobots,” in *IEEE Int. Conf. on Robotics and Automation*, 2008, pp. 886–892.

[52] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2014.

Appendix A

Differential Drive Baseline

The differential drive baseline method used in this paper is a common steering method used for robots with wheel or leg-like mechanisms on both sides. In this control scheme, the turn rate ω_{robot} is proportional to the difference between left ω_l and right ω_r leg velocities. A differential drive controller assumes that the system behavior can be thought of as two wheels connected by a common axis: Here, moving the right wheel would turn the robot to the left, and moving the left wheel would turn the robot to the right. Note that this general idea of a difference in leg velocities translating to heading change of the entire system can be implemented in many ways, and we describe our implementation below merely as a guideline.

As described in Algorithm. 1, our implementation of a differential drive controller uses robot heading, as well as perpendicular distance away from the desired path, in order to set velocity setpoints for each side of robot. In addition to standard heading control where the robot turns such that its heading matches the angle of the line, it also incorporates the perpendicular error metric to say that its heading should be more or less than the heading of the line, in order to actually move back toward the line. This controller outputs desired leg velocities at a rate of 10 Hz. To enable the realization of these leg velocities, we also implement a low-level PID controller that runs in the firmware at 1000 Hz. Encoder readings of the leg positions provide feedback, and the PID controller monitors proportional, integral, and derivative errors in order to output the PWM values required for achieving the desired leg velocities.

Algorithm 1 A Differential Drive Algorithm for Trajectory Following

```
 1: Inputs: Current state $(x, y, z, \text{roll}, \text{pitch}, \text{yaw})$,
           Desired waypoints $W = [w_0, w_1, \ldots]$,
           Controller parameters $f1$ and $f2$
 2: Line segment $L \leftarrow$ closest $[w_i, w_{i+1}]$ to $(x, y)$
 3: $d_{\text{line}} \leftarrow$ angle of $L$
 4: $p \leftarrow$ perpendicular distance of $(x, y)$ to line segment $L$
 5: if $(x, y)$ to right of $L$
 6: then $d = d_{\text{line}} + f1 * p$
 7: else $d = d_{\text{line}} - f1 * p$
 8: left leg velocity $\omega_l \leftarrow \omega_{\text{nom}} - d * f2$
 9: right leg velocity $\omega_r \leftarrow \omega_{\text{nom}} + d * f2$
10: Outputs: leg PID velocity setpoints $\omega_l$ and $\omega_r$
```

Appendix B

Choice of Action Abstraction

Our model-based learning method allows users the freedom to vary the level of abstraction at which they would like to operate. Two options, which we illustrate in Fig. B.1 as exhibiting comparable task performance, include setting direct motor PWM values and setting desired velocity setpoints.

Directly setting motor commands, instead of velocity setpoints, precludes the need to tune another layer of feedback control (i.e. lower-level PID controller) for calculating motor commands. The method of directly sending commands, however, encounters the problems involved with a lack of feedback loop. In the case of the VelociRoACH, a given PWM value can result in different amounts of leg movement, due to both variations in the battery level, as well as due to the leg kinematics leading to different forces at different stages of the leg rotation. At the same time, outputting desired velocities and then designing a lower-level PID controller to achieve those velocities involves an additional stage of parameter tuning, and one concern includes unpredictable behavior caused by not achieving the desired velocity within the time Δt before the next setpoint is received. Each of these action abstraction options has pros and cons that manifest themselves differently on different systems. Thus, it is an enticing feature to have an algorithm easily adapt to the user's choice of action abstraction.

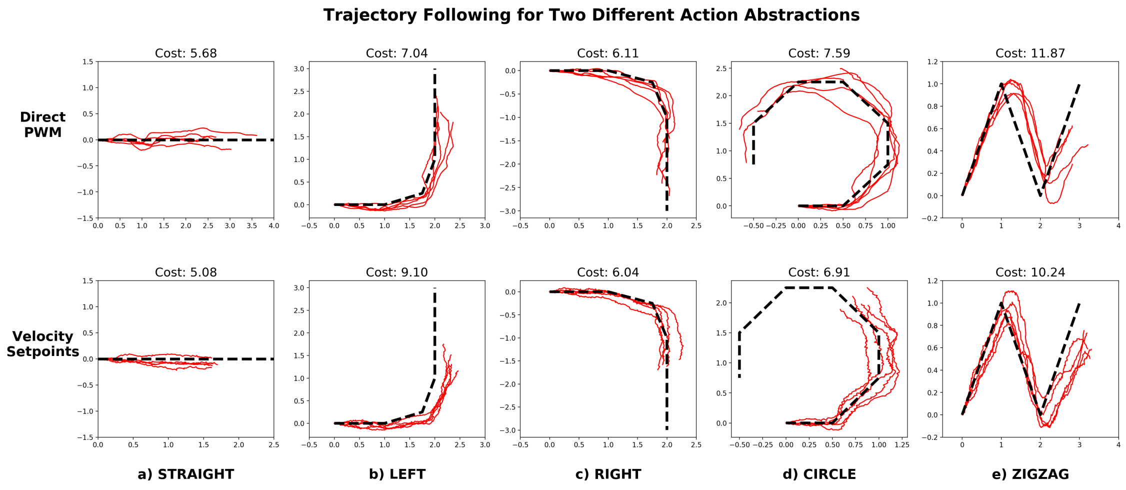


Figure B.1: Trajectories executed by the model-based controller when the control outputs were (Top:) direct motor PWM values and (Bottom:) leg velocity setpoints, which a lower-level controller was tasked with achieving. Note that for each of these options, the corresponding dynamics model was trained using data where the $\mathbf{a}_t$ represented the indicated choice of action abstraction.

Appendix C

Images of Terrain

Shown in Fig. C.1 are images of the terrain, as seen by the VelociRoACH through its 3.4 gram monocular color camera. The images include carpet, turf, styrofoam, and gravel.

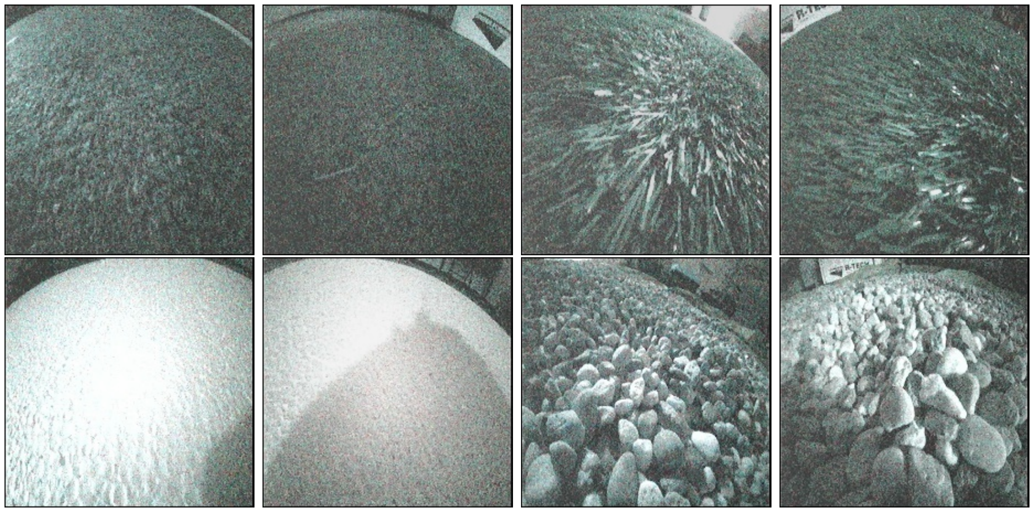


Figure C.1: Sample terrain images, as seen by the VelociRoACH.

Appendix D

Telemetry Data

The following three figures (Fig. D.1 - Fig. D.3) compare saved telemetry data from runs executed by the learned method, with data from runs executed by the differential drive strategy. Fig. D.1 shows the distribution of commanded actions for the right vs. left side of the robot during various "straight" and "left" runs. Fig. D.2 shows histograms of the roll angles during the execution of "straight," "right," and "left" trajectories. These histograms show more symmetric distributions of roll angles for the differential drive method, and more skewed roll angles for the learned method. Fig. D.3 shows 4 "left" runs from the learned method (top), and from the differential drive method (bottom). We see a clear correlation between desired robot heading and resulting control commands for the differential drive strategy, but a less clear pattern for the learned method. Finally, Fig. D.4 and Fig. D.5 show saved telemetry data (including gyro data, accelerometer data, back EMF, duty cycle, battery voltage, torque, leg positions, commanded actions, robot heading, x, y, roll, pitch, yaw) from the execution of a "left" trajectory by the two methods (differential drive method and learned method).

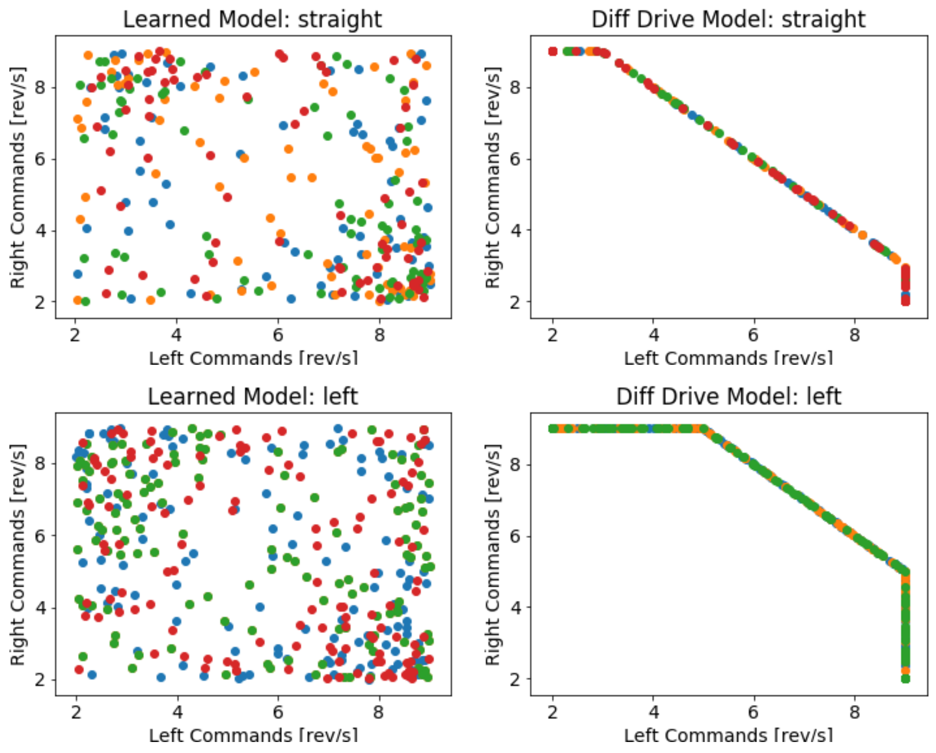


Figure D.1: Distribution of commanded actions for the right vs. left side of the robot, where the commanded actions are velocity setpoints for the legs, in units of leg revolutions per second. The top plots show multiple "straight" runs, and the bottom plots show multiple "left" runs.

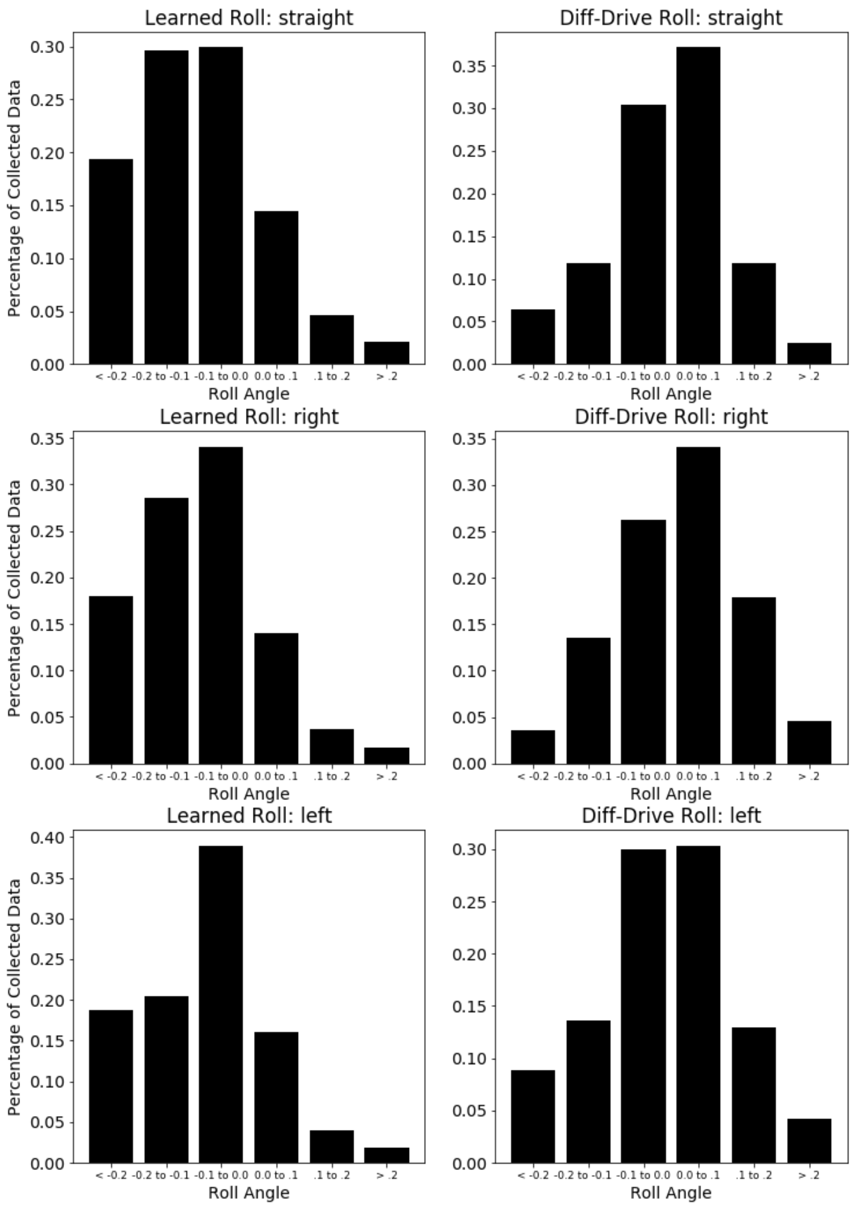


Figure D.2: Histograms of roll angles during the execution of "straight," "right," and "left" trajectories, showing more symmetric distributions of roll angles for the differential drive method and more skewed roll angles for the learned method.

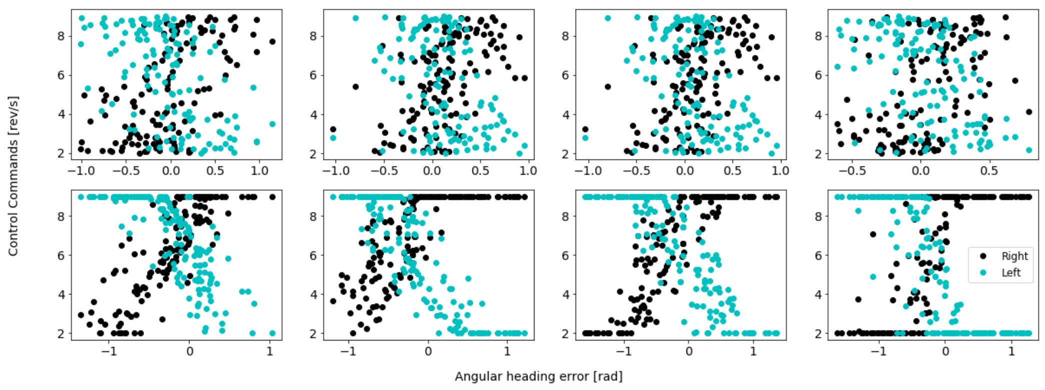
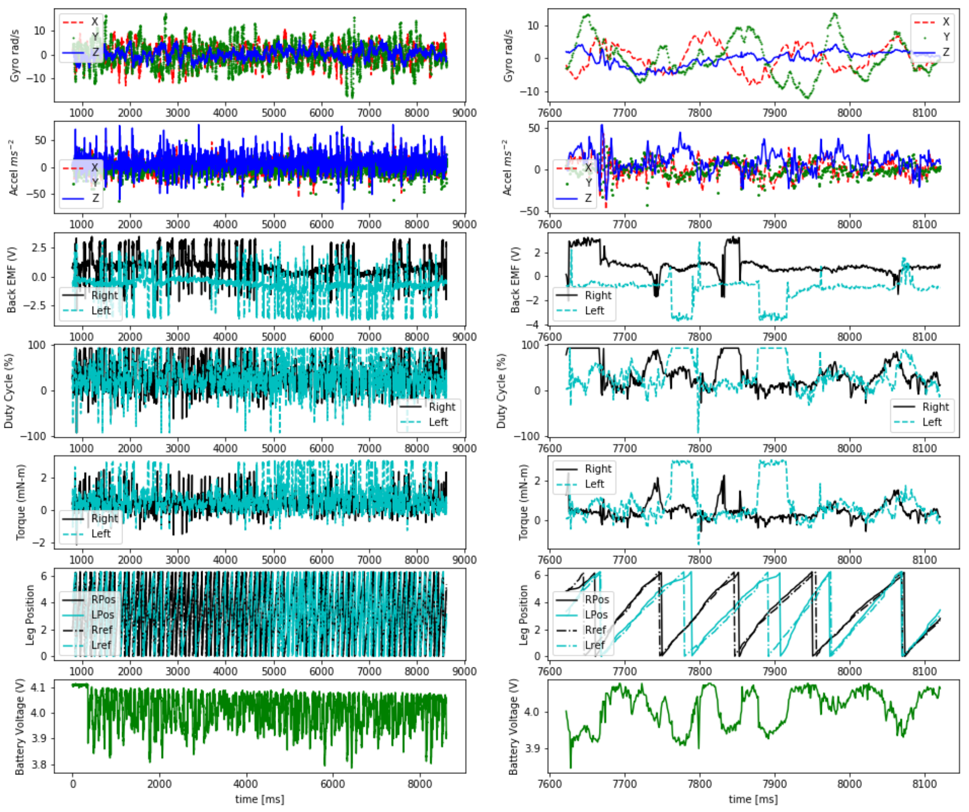


Figure D.3: Four "left" runs (top) from the learned method, and four "left" runs (bottom) from the differential drive method. The blue dots correspond to the right motor, and the left dots correspond to the left motor. The differential drive runs show a clear correlation between heading error (in radians) and resulting control commands, whereas the learned model does not. Note that the visible "noise" in the differential drive data is due to the fact that those commands depend on distance error as well as heading error.



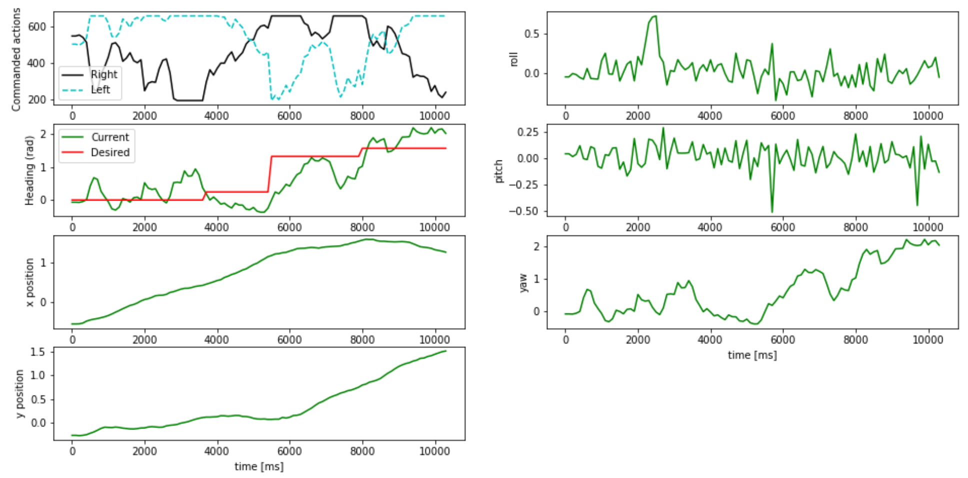


Figure D.4: **Differential Drive: left turn.** The top plots show 1kHz data from on-board the robot during a "left" run, with zoomed-in plots on the right. The bottom plots show 10Hz data from the motion capture system, during the same run. Additionally, the back EMF values (indicating motor velocities) in this figure show some rapid value changes and sometimes do not correlate to the leg positions seen, so we suspect that there is some unresolved electrical error with electronics on our circuit board, and a correction of these (occasionally) incorrect values could further help our learned method.

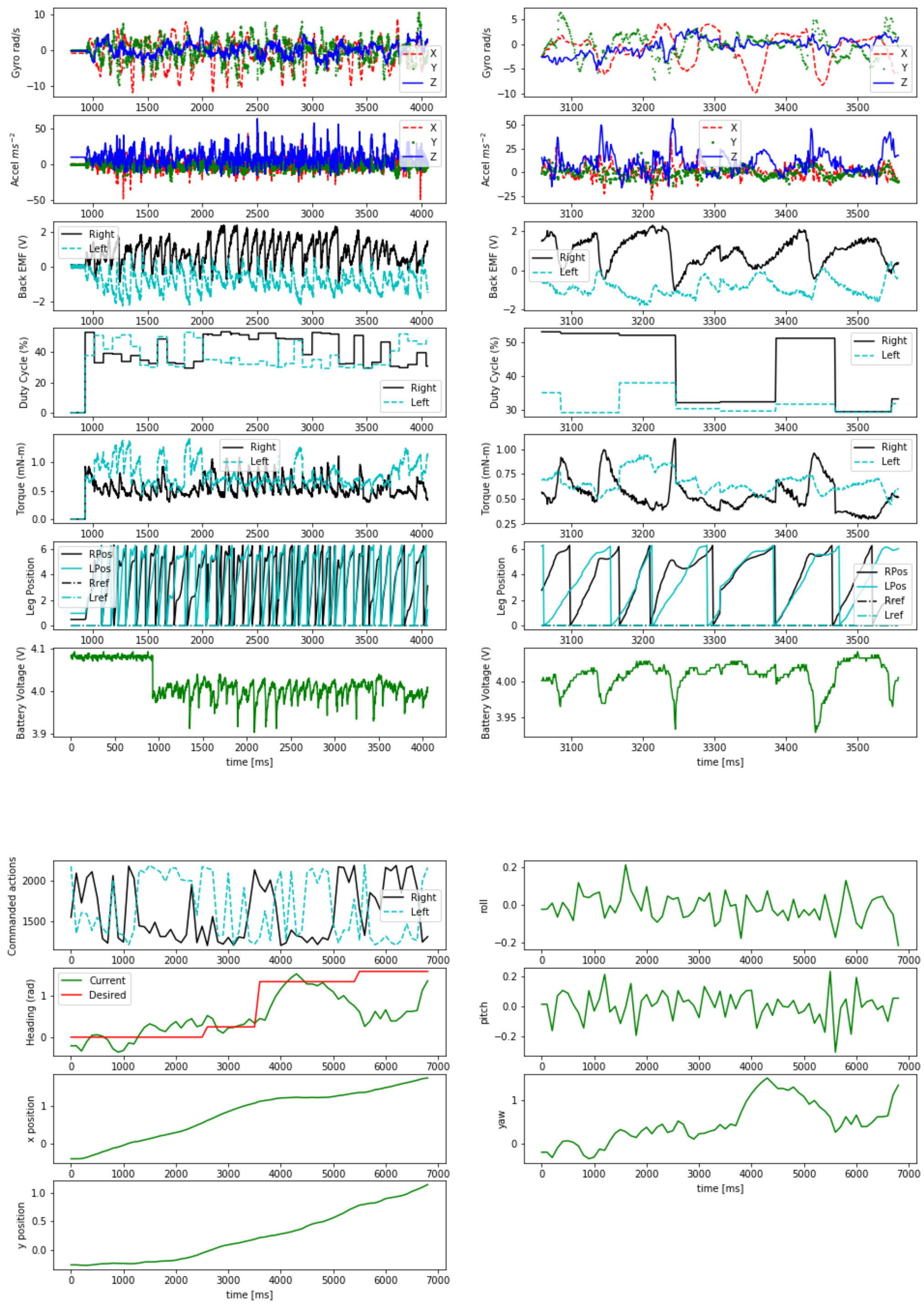


Figure D.5: **Learned Model: left turn.** The top plots show 1kHz data from on-board the robot during a "left" run, with zoomed-in plots on the right. The bottom plots show 10Hz data from the motion capture system, during the same run. Duty cycle varies less frequently here than in Fig. D.4, because our commands are PWM values sent at 10Hz (rather than velocity setpoints, which then require a controller to set PWM values at 1kHz).