Neural Network Dynamics Models for Control of Under-actuated Legged Millirobots

Anusha Nagabandi, Guangzhao Yang, Thomas Asmar, Gregory Kahn, Sergey Levine, Ronald S. Fearing
University of California, Berkeley

Abstract—Millirobots are a promising robotic platform for many applications due to their small size and low manufacturing costs. However, controlling these millirobots is difficult due to their underactuation, power constraints, and size. While hand-engineered controllers can sometimes control these millirobots, they often have difficulties with highly dynamic maneuvers and complex terrains. We present a learning based approach in which a model of the dynamics is learned from data gathered by the millirobot, and that data is then leveraged by an MPC controller. We show that with 17 minutes of random data collected with the VelociRoACH millirobot, the VelociRoACH can accurately follow trajectories at higher speeds and on more difficult terrains than a differential drive controller. Experiment videos can be found at https://youtu.be/1Wx3xThy938

I. INTRODUCTION

Small legged robots provide increased accessibility and mobility in unstructured environments, while their low power consumption and low cost enables scaling them to large teams to accomplish more complex tasks. However, modeling the hybrid dynamics of under-actuated legged robots from first principles is exceedingly difficult due to complicated ground contact physics with complex terrains. Furthermore, low-cost robots produced through rapid manufacturing techniques are likely to exhibit differing dynamics. Due to these modeling challenges, many of the most popular locomotion strategies for such systems are hand-engineered and heuristic. These manually designed controllers constrain the capabilities of these platforms, and impose a heavy burden on the engineer to devise gaits and hand-tune controllers. In this paper, we explore how learning can be used to automatically acquire locomotion strategies for small, low-cost, and highly dynamic legged robots with only minutes of real-world data.

The experimental platform in this work is the VelociRoACH, shown in Figure 1. The VelociRoACH is a 10 cm, 30 gram hexapedal millirobot capable of running at 2.7 m/s. While we can manually design controllers for this robot, these controllers are limited in their capabilities in terms of their ability to accurately follow trajectories and their generalizability to other terrains and gaits.

Choosing a learning algorithm appropriate for millirobots requires consideration of a number of factors. First, we need to be able to learn quickly from modest amounts of data, so as to make it practical to apply the algorithm to real physical robots. Second, we need to learn general-purpose models that can be used to accomplish a wide range of navigational tasks in a diverse set of environments. To that end, model-based learning presents an appealing option, in which a model of the system dynamics is first estimated from data, and then used to plan actions at test-time.

The primary contribution of our work is a model-based learning algorithm for learning locomotion gaits on dynamic legged robotic platforms that makes no assumptions about the structure of the task or the morphology of the robot. We demonstrate that multilayer neural networks can in fact learn the dynamics of the low-cost, under-actuated, and legged VelociRoACH robot using only 17 minutes of randomly generated training trajectories. We use this learned model with model predictive control (MPC) and empirically show that it is powerful enough to accomplish various locomotion tasks of following user-specified paths, exceeding the performance of a conventional differential drive controller. We further analyze our learned model by 1) demonstrating that the model encodes environmental properties and adapts to terrains, 2) showing that training the model with more data improves performance, and 3) verifying that the learning approach is agnostic to the choice of action representation.

II. RELATED WORK

The VelociRoACH platform used in this work is inspired by rapidly running arthropods like cockroaches, which can achieve remarkable locomotion performance in terms of stability, speed, and maneuverability. In light of practical considerations including cost, power, simplicity, and manufacturability, this system uses only two motors to control all six legs. This challenge of controlling under-actuated robots is prevalent in legged systems [1], [2], and much of the prior locomotion work lies in the middle ground where the systems are still underactuated, but each leg has an independent actuator. RHex [3], for example, uses a clock driven controller to achieve gaits by moving each leg as desired. For similar
quasi-static systems [4], a common control strategy includes an alternating tripod gait with asymmetrical drive of the tripods to generate turning. Unlike small and fast robots such as our VelociRoACH system, larger robots such as Anymal [5] and Big Dog [6] have the luxury of 3 degrees of freedom on each leg, as well as more time in between each step of the leg, and can thus use more sophisticated control strategies, including accurate foot placement. Other prior work with Little Dog [7], [8], [9], has combined control, planning, and learning on a system that has access to a terrain map and can be controller at the foot placement level, which is not available for the underactuated VelociRoACH system.

Various leg-based steering methods have been proposed for dynamic running robots [10], [11], such as actively changing leg kinematics [12], [13] and various dynamic turning techniques. Prior work in this area includes modulation of leg impedance to achieve high speed turning at 100 deg/sec [14], a roll oscillation modulated turning technique for millirobots achieving high speed turning gaits at 206 deg/sec [15], and a gyro-based closed-loop heading controller that uses a tail to induce transient moments to maintain turning rates of approximately 400 deg/sec. Note, however, that these steering approaches achieve turning gaits, but have not yet focused on executing accurate locomotion behaviors (i.e. fast turns as opposed to fast locomotion along a desired turning path). Furthermore, these steering methods have mostly been shown to work on flat surfaces, but may suffer from uncertain contacts and disturbances on other terrain.

Legged locomotion has been modeled using a diverse selection of reduced order templates [16]. Much prior work makes simplifying model assumptions, such as a spring loaded inverted pendulum (SLIP) model [17], [18]. Such models, including the common differential drive strategy, do succeed in certain regimes [19], but they fail at high speeds where the dynamics become more complicated, or in irregular environments where contact dynamics are complex and cannot be accurately incorporated into the model. Buchan et al. [20] presented a data-driven technique of fitting piecewise affine models to represent the dynamics of the VelociRoACH, which they used to perform forward predictions over the course of half a stride length (0.1sec) before the model diverged. In this work, we learn a dynamics model that can make reasonable predictions over a longer horizon, and we wrap a controller around that model to validate its usefulness.

Model-based learning methods, in a similar way as the modeling methods mentioned above, have used relatively simple function approximators such as time-varying linear models [21], [22]. These models have not yet been shown to succeed at complex locomotion tasks, due to the limited generalizability that rises from their low representational power. Much prior work in learning probabilistic dynamics models has used Gaussian processes [23], [24]. Deisenroth et al. [25] learn probabilistic forward models and use them to learn feedback controllers for generating walking policies on legged locomotion systems, including a real compass walker and a simulated biped.

Prior work has also presented various methods of automatic gait optimization in order to reduce the time-consuming design process of manually finding parameters for gaits and controllers; these methods include stochastic gradient descent [26], genetic algorithms [27], and Bayesian optimization [28], [29]. [26] optimizes a control policy for bipedal walking online in less than 20 minutes, on a simplified system that has 6 joints and is designed to function as a passive walker when no commands are sent. Calandra et al. [30] compare various Bayesian optimization approaches for automatic gait optimization. They demonstrate the sample efficiency of these methods and mention that future work should focus on obtaining better response surface models. While the mentioned learning methods are sample efficient and can be applied to real systems, they have not yet been shown to work for fast robots operating in highly dynamic regimes on irregular surfaces that result in complex contact dynamics.

On the other hand, deep reinforcement learning algorithms, such as Q-learning [31], [32], actor-critic methods [33], [34], and policy gradients [35] have learned complex skills in high-dimensional state spaces, including skills for simulated robotic locomotion tasks. The high sample complexity of such purely model-free algorithms, however, makes them difficult to use for learning in the real world, where sample collection is limited by the constraints of real-time operation. Although no prior method has attempted model-free deep reinforcement learning of locomotion skills in the real-world to our knowledge, Gu et al. [36] report learning reaching skills with a robotic arm using several hours of experience. Our method can learn models that are suitable for real-world locomotion using only minutes of experience, and can then repurpose this model for new locomotion skills that were not attempted during data collection.

In our prior work on model-based reinforcement learning [37], we studied how neural network models can enable sample-efficient learning of locomotion skills for simulated benchmark tasks, through a combination of model-based learning and model-free fine-tuning. In this work, we show how model-based learning with neural network dynamics models can be extended to a real-world robotic platform, illustrate a variety of waypoint following skills on different surfaces, and compare to a commonly used control strategy based on differential drive, illustrating considerable improvement resulting from learning system dynamics.

III. SYSTEM DESCRIPTION

The VelociRoACH is a minimally actuated, small, legged, and highly dynamic palm-sized robotic platform. It is 10 cm in length, approximately 30 grams in weight, can move up to 27 body-lengths per second, and uses two motors to control all six legs.

A. Mechanical Design

Several design features on the VelociRoACH aim to attain the dynamic performance of its biological inspiration, the American cockroach Periplaneta americana. Details of design techniques and parameter selection can be found in the
origin VelociRoACH paper [38], but we provide relevant information below.

1) Structure: The smart composite microstructure (SCM) process [39] allows for the creation of lightweight linkages, enabling the rapid realization of fully functional prototypes of folded flexure-based mobile millirobots. The process utilizes low cost materials to create, in less than one hour, a robot chassis for under $2 per robot. The general SCM process can be thought of as creating flexures by embedding a layer of flexible material in between layers of cut-out rigid material.

Unlike the original VelociRoACH [38], which used structural material of 4 ply cardboard and a flexural material of 1 mil (0.001") polyethylene terephthalate (PET), this new design has a structural material of 10 mil PET and a composite flexural material of 1.1 oz ripstop nylon. The nylon is also bonded to 1 mil PET, for added structural strength. The layers are bonded together with sheets of hot mount adhesive (GBC Octiva) using a laminator with a roller temperature of 315 F.

An overview of our steps for constructing the robot chassis is as follows: laser cut sheets of 10 mil PET for the structural material, laminate a sheet of nylon and 1 mil PET for the flexural material, laminate the nylon between two 10 mil PET sheets using thermal adhesive, laser cut the full workpiece, and then assemble the body pieces to form the robot chassis. The ‘C’ shaped legs are cast using urethane plastic (Innovative Polymers TP-4014) and allows the creation of legs with varying stiffness.

The VelociRoACH’s rigid structural core houses the battery, motors, transmission, microcontroller, and all sensors. The core also provides mechanical grounding points for the kinematic linkages, whose geometry was designed similarly to the Hoover et al. linkage design for the DynaRoACH robot [14]. Each side of the robot has one motor and linkages (including four-bar mechanisms) which translate the rotary motion from the motor into useful leg movements. These linkages also couple a single motor to multiple legs, reducing the number of actuators required.

2) Motor Transmission: The robot’s legs are driven by a custom transmission that independently drives the left and right sets of legs. Each transmission side is driven by a 3.6 ohm DC motor with a 21.3:1 gear reduction. As shown in Fig. 2, the fore and aft legs are constrained to be 180° out of phase from the middle leg. Similar to the design of the X2-VelociRoACH [40], this VelociRoACH uses two connected output cranks per side to transmit forces at each leg contact. This is in contrast to previous VelociRoACH designs, which use a single output crank per side with a planarizing SCM four-bar linkage to drive the legs. This improved transmission design maintains the leg trajectories of the robot’s kinematic linkages even under an added payload of up to 30 grams.

B. Electronics

The VelociRoACH carries an ImageProc embedded circuit board [41] 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 add 14-bit magnetic rotary encoders to the motors on each side of the robot to monitor absolute position. Additional sensory feedback includes information such as battery voltage and back-EMF signals from the motors. Onboard processing is done in a 1 kHz control loop that is written in the firmware for the microprocessor. However, due to computational limits of the microprocessor, we stream data from the robot to a laptop (at 10-60 Hz, depending on data as well as application) for more intensive computations, and then stream commands back to the microprocessor for the firmware to implement. Finally, to bypass the problem of using only on-board sensors for state estimation, an OptiTrack motion capture system streams robot pose information during experiments.

IV. DIFFERENTIAL DRIVE BASELINE METHOD

A common steering method used for robots with wheel or leg-like mechanisms on both sides is known as differential drive, where the turn rate \( \omega_{\text{robot}} \) is proportional to the difference between left \( \omega_l \) and right \( \omega_r \) 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

https://github.com/biomimetics/imageproc_pcb
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 \((x, y, z, \text{roll, pitch, yaw})\), 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 \(d = d_{\text{line}} + f1 \ast p\)
7. else \(d = d_{\text{line}} - f1 \ast p\)
8. left leg velocity \(\omega_l \leftarrow \omega_{\text{nom}} - d \ast f2\)
9. right leg velocity \(\omega_r \leftarrow \omega_{\text{nom}} + d \ast f2\)
10. **Outputs:** leg PID velocity setpoints \(\omega_l\) and \(\omega_r\)

V. Locomotion Control with Neural Network Dynamics Models

As described above, the VelociRoACH is designed to be a rapidly manufactured low-cost legged platform that can be used in complex environments. We seek an automated method of acquiring locomotion strategies; however, its high speeds of operation, partially observable state space, and under-actuated design make data-driven modeling and control challenging.

A. Model-Based Learning Method

We describe below a method for learning a dynamics model and using the learned model as part of a model predictive control (MPC) algorithm for the real VelociRoACH robotic platform. We discuss our learned dynamics function in Sec. V-A.1 how to train the learned dynamics function in Sec. V-A.2 how to use that learned dynamics function to implement a controller in Sec. V-A.3 and a cost function that we use for trajectory following tasks in Sec. V-A.4

1) Neural Network Dynamics Function: The learned dynamics function \(f_\theta(s_t, a_t)\) is parameterized as a multilayer neural network, where the parameter vector \(\theta\) represents the weights of the network. This function takes as input the current state \(s_t\) and action \(a_t\), and outputs the predicted change in state \(s_\text{pred}\) that occurs over the time step duration of \(\Delta t\). Thus, the predicted next state is as follows: \(s_{t+1} = s_t + f_\theta(s_t, a_t)\).

Increasing this \(\Delta t\) increases the information represented by each data point, which can help with dynamics learning as well as with planning using the learned dynamics model (Sec. V-A.3), because we can plan ahead for a given amount of time using fewer open-loop prediction steps. While having too small of a \(\Delta t\) leads to too small of a state difference to allow meaningful learning, increasing the \(\Delta t\) 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 \(\Delta t\) values for our system, we provide this insight as something for consideration when implementing this method on other systems.

We define the state \(s_t\) of the VelociRoACH to be \([x, y, z, v_x, v_y, v_z, \cos(\phi), \sin(\phi), \cos(\phi_p), \sin(\phi_p), \cos(\phi_y), \sin(\phi_y), \omega_x, \omega_y, \omega_z, \cos(\alpha L), \sin(\alpha L), \cos(\alpha R), \sin(\alpha R), v_aL, v_aR, \text{bemf}_L, \text{bemf}_R, V_{bat}]^T\). The center of mass positions \((x, y, z)\) and the Euler angles to describe the center of mass pose \((\phi, \phi_p, \phi_y)\) 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 \((\alpha L, \alpha R)\) come from the magnetic rotary encoders, which give a notion of leg position. We include \((\text{bemf}_L, \text{bemf}_R)\) because back-EMF provides a notion of motor torque/velocity, and \((V_{bat})\) because the voltage of the battery does affect 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 not encounter wrapping issues.

Finally, we have multiple options for the action \(a_t\) representation, and we mention two in this work. The first is to specify pulse width modulation (PWM) values to the motor, which can be thought of as specifying motor torques. The second is to specify velocity setpoints for the motor, and have an underlying PID controller to achieve these velocities. We discuss these action abstractions in Sec. VI-D and consider the advantages and disadvantages of each.

2) Training the Dynamics Function: We collect training data on the VelociRoACH by placing the robot in an arbitrary start state and executing random actions at each time step. We record each resulting trajectory \(\tau = (s_0, a_0, \cdots, s_{T-2}, a_{T-2}, s_{T-1})\) of length \(T\). We slice the trajectories \(\{\tau\}\) into training data inputs \((s_t, a_t)\) and corresponding output labels \((s_{t+1} - s_t)\). We train the dynamics model \(f_\theta(s_t, a_t)\) on data from the training dataset \(\mathcal{D}\) by minimizing the error

\[
E(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(s_t, a_t, s_{t+1}) \in \mathcal{D}} \frac{1}{2} \| (s_{t+1} - s_t) - f_\theta(s_t, a_t) \|^2
\]

using stochastic gradient descent. Prior to training, we preprocess the training data by normalizing the data to be mean 0 and standard deviation 1, which ensures that the loss function weights different parts of the state equally, regardless of their magnitudes.

3) Model-Based Control with Learned Dynamics: We formulate a model-based controller which uses the learned model \(f_\theta(s_t, a_t)\) together with a cost function \(c(s_t, a_t)\) that encodes some task. Many methods could be used to perform this action selection, and we use a random-sampling shooting method [41]. 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 future states from executing each of those sequences, and then use the cost function \(c(s_t, a_t)\) to select the optimal action sequence. Rather than executing this entire selected sequence of actions \(A^{(H)} = (a_t, \cdots, a_{t+H-1})\), we use model predictive control (MPC) to execute only the first action \(a_t\), and then replan at the next time step, given updated state information.
4) Cost Function for Trajectory Following: As mentioned above, we specify a task in the form of a cost function $c(s_t, a_t)$, which is used to evaluate each of the predicted trajectories that result from the $K$ candidate action sequences, before selecting the most desirable one. Here, we describe a cost function that allows the robot to follow a trajectory that is specified by sparse waypoints that represent desired $(x, y)$ center-of-mass positions. Our cost function $c(s_t, a_t) = f_p * p + f_h * h + f_f * f$ consists of a parameter $f_p$ to penalize the perpendicular distance $p$ away from the desired path, a parameter $f_f$ to encourage forward progress $f$ along the path, and a parameter $f_h$ to keep the heading $h$ of the system toward the desired direction.

VI. 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. To provide a comparison of the algorithm’s performance, we use the differential drive approach described in Section IV as a baseline method.

We implement the learned dynamics function $f_θ(s_t, a_t)$ as a multilayer neural network model consisting of 2 hidden layers, each of dimension 500, with ReLU activations. For all experiments and results reported below, we use only 17 minutes (10,000 datapoints) of data to train the dynamics model: We collect 200 rollouts, each containing 50 data points that are collected at 10 Hz. We train the dynamics model for 75 epochs, using the Adam optimizer [42] 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 $N = 1000$, the amount of time represented by one time step $Δt = 0.1$ sec, the horizon $H = 4$, and parameters $f_p = 40, f_f = 10$, and $f_h = 15$ for the perpendicular, forward, and heading components of the cost function from Sec. V-A.4 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.

For our baseline implementation of a differential drive steering method, we use $f1 = 2$ and $f2 = 5$ for the weights mentioned in Alg. 1 and we use $p = 1800, i = 200$, and $d = 100$ for the proportional, integral, and derivative gains in our low-level leg-position PID controller. All cost numbers reported below are calculated on the same cost function (from Sec. V-A.4) that indicates how well the executed trajectory aligns with the desired trajectory, and each reported number represents an average over 10 runs.

Note that the training data is gathered entirely using random trajectories, and therefore the trajectories 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. Furthermore, although the model is trained only once, we use it to accomplish a variety of tasks at run-time by simply changing the cost function. This eliminates the need for task-specific training, which further improves sample efficiency. We show in Fig. 3[3] some images of the VelociRoACH executing right, straight, and left trajectories using our model-based learning method.

A. Comparing to Differential Drive

In Fig. 4 we see that a differential drive control strategy succeeds at following desired trajectories when operating at a low speed. However, as leg speeds increase, 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 [15]. 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 illustrates, on different trajectories, that the two methods are comparable at low speeds. However, our model-based approach outperforms the differential drive strategy at higher speeds, thus suggesting that we learned something that was not merely a differential drive steering strategy.

B. Changing Surfaces

To verify whether our learned model encapsulates information about the environment, we conducted experiments on a carpet material and a styrofoam material which was slippery and had no compliance. Fig. 5a shows the average cost incurred during the task of trajectory following, evaluated for both controllers, for different trajectories, and on different surface materials. We see that our model-based controller outperforms the differential drive baseline in each of the conditions. Furthermore, although differences in leg contacts alter system dynamics and cause the performance of differential drive to deteriorate on a slippery surface, the model-based controller is able to maintain comparable performance on these two very different surfaces. Thus, we can infer that the learned dynamics model learns something about the surface.

Fig. 5b illustrates the $(x, y)$ trajectories taken during the
Fig. 4: 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.

Table I 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 the differences in those surfaces and suggests that our learned dynamics model does in fact encode some knowledge about the surface. Relatedly, performance diminishes when the model is trained on data gathered from both terrains, which likely indicates that more work is needed to develop algorithms for learning models that are effective across various task settings.

C. Improving Performance with More Data

To explore the effect of the quantity of training data, we trained three different dynamics models using different amounts of training data. We trained one with 50 rollouts (4 minutes), one with 200 rollouts (17 minutes), and one with 400 rollouts (32 minutes). Table II indicates that more training data can indeed increase task performance. This is an encouraging indication that improvement can occur over time, which is not the case for hand-engineered solutions.
Fig. 6: Trajectories executed by our 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 \( a_t \) represented the indicated choice of action abstraction.

D. 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. 6 as exhibiting comparable task performance, include setting direct motor PWM values and setting desired velocity setpoints.

Directly setting motor commands 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 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 \( \Delta t \) 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.

VII. Discussion

We presented a model-based learning algorithm based on neural network dynamics models that enables accurate locomotion of a low-cost, under-actuated, legged, and highly dynamic VelociRoACH robot on various surfaces. Using only 17 minutes of real-world data, our method outperformed a commonly used differential drive method, particularly at higher speeds. Our experiments also indicate that the model learned to encode environmental information, and that training with more data can improve performance.

We used our sample efficient model-based method to automatically acquire locomotion strategies for a complex and real robotic system, with no need to hand-engineer a controller. Also, we learned a dynamics model which can be repurposed at run-time to execute different trajectories through changing the cost function, thus eliminating the need for task-specific training. While this is an appealing advantage over, for example, a policy trained to accomplish a certain task, one drawback of such a controller 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 better 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 the need for 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.

An improvement to our method would be to develop an algorithm for online adaptation of the model, especially because over time, the roach suffers from deterioration of chassis durability, motor strength, and leg characteristics. 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. We currently sample random actions at each time step and select one based on the predictions of our dynamics model. However, this is not feasible on systems with a high-dimensional action space because we would need to sample too many candidate action sequences in the hope of finding a good one. Furthermore,
our method does not currently apply any prior knowledge or impose any structure on the actions, which could drastically improve performance. 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.

VIII. ACKNOWLEDGEMENTS

We would like to thank Carlos Casarez for 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.

REFERENCES

[1] M. D. Berkemeier and R. S. Fearing, “Sliding and hopping gaits for the underactuated acrobot,” IEEE Transactions on Robotics and Automation, vol. 14, no. 4, pp. 629–634, 1998.

[2] K. Byl, “Metastable legged-robot locomotion,” Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[3] 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.

[4] A. M. Hoover, E. Steltz, and R. S. Fearing, “Roach: An autonomous 2.4 g crawling hexapod robot,” in IEEE/RSJ Int. Conf. on Intell. Robots and Systems, 2008, pp. 26–33.

[5] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsoinis, J. Hwangbo, B. Kode, P. Fanahausser, M. Bloesch, et al., “Anymal-a highly mobile and dynamic quadruped robot,” in IEEE/RSJ Int. Conf. on Intell. Robots and Systems, 2016, pp. 38–44.

[6] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, “Bigdog, the rough-terrain quadruped robot,” in HAC Proceedings Volumes, vol. 41, no. 2, pp. 10 822–10 825, 2008.

[7] 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.

[8] 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.

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

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, “The dynamics of legged locomotion: Models, analyses, and challenges,” SIAM Review, vol. 48, no. 2, pp. 207–304, 2006.

[17] H. Komsuoglu, K. Sohn, R. J. Full, and D. E. Koditschek, “A physical model for dynamical arthropod running on level ground,” Departmental Papers (ESE), p. 466, 2008.

[18] H. Komsuoglu, A. Majumdar, Y. O. Aydin, and D. E. Koditschek, “Characterization of dynamic behaviors in a hexaped robot,” in Experimental Robotics. Springer, 2014, pp. 667–684.

[19] 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.

[20] 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.

[21] A. Lioutkiv, A. Paraschos, J. Peters, and G. Neumann, “Sample-based information-theoretic stochastic optimal control” in IEEE Int. Conf. on Robotics and Automation, 2014.

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

[23] M. Deisenroth and C. Rasmussen, “A model-based and data-efficient approach to policy search” in ICML, 2011.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, “Automatic gait optimization with gaussian process regression.” in IJCAI, vol. 7, 2007, pp. 94–949.

[29] 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.

[30] 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.

[31] V. Mnih and et al., “Human-level control through deep reinforcement learning.” Nature, 2015.

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

[33] T. P. Lillicrap and et. al., “Continuous control with deep reinforcement learning.” ICLR, 2016.

[34] 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.

[35] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in ICML, 2015.

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

[37] 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.

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

[39] 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.

[40] D. W. Haldane and R. S. Fearing, “Running beyond the bio-inspired regime,” in IEEE Int. Conf. on Robotics and Automation, 2015, pp. 4539–4546.

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

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