Learning Legged Swimming Gaits from Experience

David Meger, Juan Camilo Gamboa Higuera, Anqi Xu and Gregory Dudek

***Abstract*— We present an end-to-end framework for realizing fully automated gait learning for a complex underwater legged robot. Using this framework, we demonstrate that a hexapod flipper-propelled robot can learn task-specific control policies purely from experience data. Our method couples a modern policy search technique with a family of periodic low-level controls that are well suited for underwater propulsion. Our experimental results demonstrate the practical efficacy of *tabula rasa* learning of parameterized policies that actuate the six legs of a hexapod swimmer to successfully carry out a variety of acrobatic maneuvers in SO(3). We also demonstrate *informed* learning, which allows a small amount of human oversight to bootstrap the process. In numerous cases, novel emergent gait behaviors have arisen from learning, such as the use of one stationary flipper to create drag while another oscillates to create thrust. Similar effective results have been demonstrated in under-actuated configurations, where as few as two flippers are used to maneuver the robot to achieve both SO(3) angles and angular rates task targets. The success of our learning framework is assessed both in simulation and in the field using a physical underwater swimming robot.**

I. INTRODUCTION

In this paper, we study the task of learning swimming controllers for the six hydrofoils (alternatively: flippers or fins) of the Aqua family of hexapod amphibious robots [1]. The task of coordinating the motion of multiple legs for swimming is challenging due to high intrinsic dimensionality and complexities that arise from hydrodynamics. For example, the force generated by an oscillating hydrofoil is due to *Karman street* flow pattern, which are known to be difficult to model [2], [3], [4]. Recent progress in learning gaits for terrestrial legged robots has demonstrated the benefits of data-driven approaches, particularly for system aspects that are difficult to model directly [5], [6], [7].

Our work represents the first attempt at a fully data-driven approach for leg control of Aqua, which contrasts with expert-engineered swimming gaits [3] and motion controllers [8] previously developed for this family of underwater robots. The learned controllers produced by our system (e.g., Figure 1) demonstrate effective swimming in a number of novel ways not previously seen on this vehicle. For instance, our learned policies dynamically alter the use of individual flippers between stationary dragging and active propulsion, presumably to maximize task efficiency. Various combinations of elementary motions have been observed in policies for the same high-level task. For example, the task "U-turn", has been executed by remaining flat (similar to a car), with a moderate bank (similar to an airplane), or by rolling completely sideways (unlike any common vehicle). Furthermore, even when using a highly underactuated two-flipper configuration, we have demonstrated successful learning of multiple acrobatic maneuvers in the unconstrained underwater domain, which is a first for this family of robots.

Mobile Robotics Laboratory, School of Computer Science, McGill University, Montreal, Canada {dmeger,gamboa,anqixu,dudek}@cim.mcgill.ca

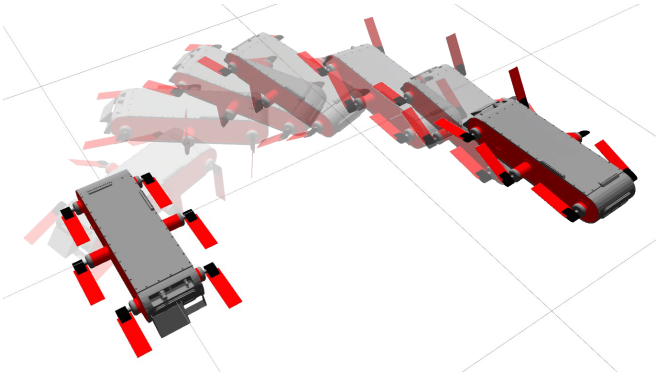


Fig. 1. A learned policy for a 6 flipper swimming task is visualized using the Gazebo simulation environment developed to aid learning. The task learned was a 180° yaw (U-turn).

Our gait learning approach is based on a policy search technique named PILCO (probabilistic inference for learning control) [9], which has recently been successfully applied to many practical robotic systems. PILCO is capable of optimizing control policies without specific knowledge of the system dynamics, through the use of experience data. However, both the practical learning performance and also its computational cost method are highly dependent on the task properties. To facilitate successful learning of acrobatic maneuvers with our complex underwater robot, we have exposed its leg control interface through an intuitive yet powerful family of low-level motor controllers for swimming, which we call *periodic leg commands* (PLC). We investigate both *tabula rasa* learning, without the use of any prior system knowledge, and also *informed* learning, which is endowed with some approximate system knowledge available from a physics simulation of our platform. Additionally, our implementation makes use of cloud-based computing to learn tasks in parallel. This enables pratical learning of many tasks in the field.

Following a brief survey of related work, Sec. III describes our reinforcement learning (RL) solution for *tabula rasa* learning of swimming controllers. Sec. IV presents our experimental procedure. Empirical results of our method, shown both in simulation and on the physical Aqua [1] amphibious platform are described in Sec. V.

II. BACKGROUND

A. Gait Learning

Gait optimization and discovery has a long history. Terzopoulos *et al.* considered not only legged walking with simulated muscle actuators, but also crawling and even swimming, albeit for idealized entities and settings [10]. Several authors have considered gait learning for physical robots operating on land [5], [6], [11], [12], [7]. Relatively few authors have considered swimming gaits for real robots, with [13] being an exception, albeit for a highly specialized robot platform. Although not strictly a learning technique, one standard approach to gait synthesis in the water, as well as on land, is the central pattern generator that can be used to produce a constrained family of gaits [14]. Most of these existing results address either gaits with limited maneuverability or vehicles with limited performance characteristics. In addition, the availability of ground truth for the assessment is a consistent challenge for maneuverable underwater vehicles.

In the experiments below, we evaluate the use of a simulator to bootstrap learning on a real robot. This choice is inspired by authors such as Xu *et al.* [15], [16] who have recently shown that human guidance, in the forms of demonstrations or direct participation, are effective aids for learning practical robotic tasks. Additionally, numerous authors [17], [18], [19] suggest benefits from transfer of information both forward and backwards between the learning procedures of the simulated and real robots. Abbeel *et al.* [18] notably investigated the use of an inaccurate analytical model to facilitate policy search approaches.

B. PILCO

The PILCO method, developed by Deisenroth *et al.* [9] has been capable of learning controllers for systems ranging from 1D inverted pendula to unicycles and manipulators. It is a promising choice for RL on physical platforms given its natural handling of continuous states and controls, as well as the ability to cope with high state dimensionality. As PILCO is a core component of our system, we will briefly describe the method details here for completeness.

As with any policy search method, PILCO aims to determine policy parameters θ^* that optimize task performance measured as the expected loss over fixed-length episodes, $\sum_{t=0}^{T} \mathbb{E}\left[L(x_t)\right]$. $L(x)$ is a loss function provided by the system designer to specify the target task. Policy search requires parameterized policies $u = \pi(x, \theta)$, capable of generating commands u based on state x. Countless policy families are possible and we will describe our choices for swimming below. PILCO is a model-based method, however it does not require prior system knowledge to be provided. Instead, data is used to learn a forward dynamics model. A Gaussian Process (GP) is used to predict the distribution of state changes $\Delta_t = (x_{t+1} - x_t)$ based on previous state and control action: $\Delta_t \sim GP(x_t, u_t)$.

PILCO estimates state (equivalently: loss) trajectories using an analytical approximation of the state distribution and its derivatives after rolling out a policy for T timesteps. This allows for gradient descent over θ to evaluate numerous potential parameters without the need of additional experience gathering on the target system. PILCO achieves excellent data efficiency relative to existing methods. The computational cost of the method scales with the amount of experience, which has motivated our development of the coarsely sampled PLC control-space rather than direct position control of our robot's actuators.

III. GAIT LEARNING METHOD

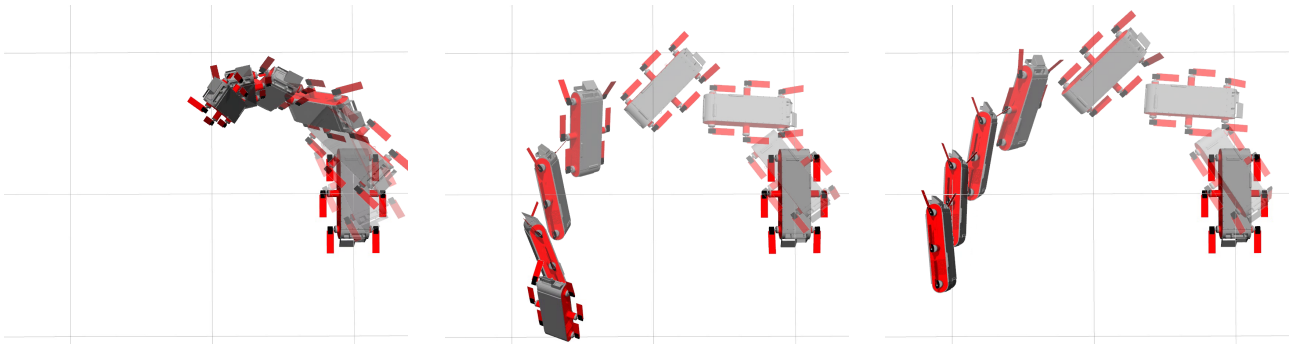
This section describes our end-to-end system for gait learning. We implement episodic RL on our robot by coupling its sensors and actuators to an RL control module. This module accepts tasks from the user in the form of loss functions. The system is initially not equipped to perform well on these tasks, as we disable all sensor-based control and coordination modules, only providing the lowest-level motor drivers that implement our parameterized PLC commands. The robot executes a sequence of policies, which generate a PLC command based on the sensory state at every timestep. Over multiple task episodes, experience data is used to learn updated control policies for the task, and performance improves to convergence. This section will continue by describing each of the elements of our system in detail.

A. Swimming State Space

Our robot senses its state using an inertial measurement unit (IMU), a pressure sensor to measure depth below the water's surface and motor encoders to determine leg positions. The data from these sensors forms the state-space, which determines x at each timestep. Notably, this is also the space over which the user is able to define the loss (negative reward) function, $L(x)$. We have investigated two types of loss functions in order to build up a dynamic set of motions for our robot: *static-angles static-depth* tasks ask the robot to achieve and stably maintain a target orientation; and *dynamic-angle static-depth* tasks require the robot to regulate its angular trajectory, typically following a smooth path such as a corkscrew. For the purposes of this work, we do not consider positional feedback, but this could potentially be added for a swimming robot equipped with an external or sensor-based localization module.

B. Periodic Leg Commands

As mentioned above, accurate modeling of the higher-order dynamics of legged swimming platforms remains an intractable problem given the fluid dynamics involved. However, a number of heuristically useful low-level swimming motions have been determined, through biological inspiration [20] or other modeling techniques. These motions include oscillation, spinning, and braking by statically opposing water flow. In order to maintain a tractable learning problem complexity, we expose a naturally useful set of low-level controls to the learning mechanism.



(a) Episode 1 (b) Episode 5 (c) Episode 10

Fig. 2. Three iterations of learning task 6 on the simulator. The task is a 180° yaw and 90° roll, known as "U-turn and knife-edge". The robot's initial policy (left) does not progress towards the goal. By learning iteration 5 (middle), the robot has reached the goal but overshoots slightly and loses roll stability. At iteration 10 (right) and beyond, the robot stably executes the motion – note that this task has been learned using only two out of six flippers for propulsion. Note the robot starts each trial from the right-most pose in each panel.

Specifically, we have developed a parameterized low-level leg controller capable of producing a broad family of per-leg motions that is general enough to allow variation to different tasks, while allowing candidate solutions to be represented succinctly. Periodic leg commands (PLC) encode position for leg i, ϕ_i, as a function of time t, and command parameters u_i: $\phi_i = PLC(t, u_i)$. The PLC function is a sinusoid, with form

$$PLC(t,u) = A \cdot sin(2\pi ft + \varphi) + b \tag{1}$$

where each command parameter vector contains amplitude, A, frequency, f, phase, φ, and angular offset, b. Our low-level leg controller computes ϕ_i, and actuates leg motion via feedback control on the motor encoders at a rate of 1 kHz. The periodic nature of the PLC command space means that a single u vector will keep the legs continuously in motion, until the next command is received. This allows policies to be productively learned at a coarse time scale, saving significant computational cost.

C. RBF Policies

We require a policy family that is able to map states into PLC actions, and which is parameterized to allow PILCO to perform loss optimization. Inspired by recent success in RL for continuous state and action spaces, we adopt radial basis function policies, which take the form

$$\pi(x,\theta) = \sum_{i=1}^{n} t_i \phi_i(x) \tag{2}$$

$$\phi_i(x) = exp(-\frac{1}{2}(x-c_i)^T \Lambda^{-1}(x-c_i)) \tag{3}$$

The parameters of each RBF component represent its center, c_i, in the sensor state space, and target, t_i, in the output command space (one PLC per active flipper). Intuitively, the RBF policy smoothly interpolates targets based on a weighted distance of the state x from each center.

IV. EXPERIMENTAL METHODOLOGY

This section describes our approach for evaluating gait learning performance on a number of sample swimming tasks. We have implemented a realistic physics simulator based on Gazebo, which enables exhaustive testing and validation prior to deployment on real hardware. Learning with PILCO has a high computational cost, and so we have implemented a cloud-based parallelization method in order to effectively learn multiple tasks within a reasonable time-frame. We perform both *tabula rasa* as well as *informed* learning, which utilizes information transferred from the simulator.

This section will describe the common experimental procedures used in each case before the following section discusses the results.

A. Sample Tasks

We have explored a broad range of gait learning tasks, which can be seen in our multi-media attachment as well as on our project page[1]. One sample task is displayed in Figure 2. For the purposes of quantitative comparison, we restrict our focus to the seven fixed-depth targets listed here:

1) flat 180° yaw (U-turn)
2) straight-ahead 90° roll (knife-edge)
3) straight-ahead 180° roll (belly-up)
4) straight-ahead clockwise rapid roll change (fast corkscrew)
5) straight-ahead anti-clockwise slow roll change (slow corkscrew)
6) 180° yaw and 90° roll (knife-edge plus U-turn)
7) 180° yaw and 180° roll (belly-up plus U-turn)

The previous human-engineered control system on Aqua is able to produce each of these motions using six flippers, if sometimes clumsily. To increase the challenge for our data driven method, we have restricted the robot to swimming with only its back two flippers. This highly underactuated

[1]Address: `http://www.cim.mcgill.ca/~dmeger/ICRA2015_GaitLearning/`

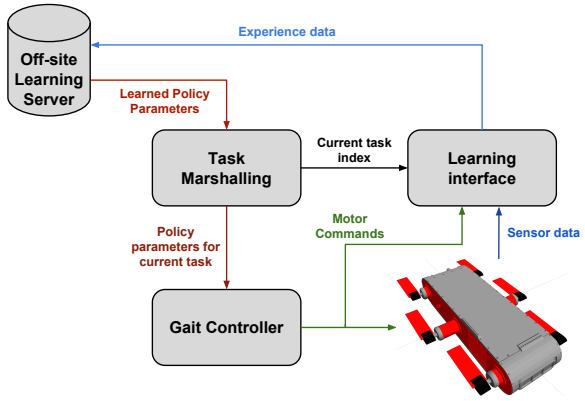


Fig. 3. A visual description of the basic elements in our parallelized off-board policy search system.

scenario requires precise control and coordination of the two available control surfaces.

B. Parallelized Off-board Policy Search

Learning with PILCO is many times slower than real-time for the swimming tasks that we consider and typical parameterizations and choices of desktop workstation architecture. This presents a practical problem, as the time running experiments on the physical robot is expensive, in terms of power consumption and human labour. To minimize the robot's idle time, and best use the available resources, we run multiple learning jobs in parallel on a cloud-compute server. We achieve this through a data marshaling architecture, depicted in Figure 3. As an episode finishes, experience data is transferred offsite, a remote learning job is instantiated and the robot moves on to learning a subsequent task. When the remote learning job is completed, the resulting policy is copied back to the robot and the next episode for that learning task is entered into a queue. We have successfully learned up to 20 tasks in parallel with this interface, allowing our robot to run nearly continuously, despite the long learning cycle of each task.

C. Swimming Simulator

We have developed a simulation environment for the Aqua robot operating underwater. Our implementation includes a set of plugins for the Gazebo simulation environment [21]. While Gazebo provides dynamics simulation using existing physics engines, such as ODE or Bullet, it does not simulate hydrodynamics or thrust models for underwater vehicles. Similar to previous work in quadrotor UAV simulation [22], our plugins augment the simulation environment with a hydrodynamics model, a thruster model, and a module that emulates interactions with the Aqua hardware.

We assume a static fluid and approximate the shape of the Aqua robot body with a rectangular prism. We consider only linear and rotational drag, added mass, buoyancy and gravity, following the description of these effects by Georgiades [2]. Each leg is assumed to be a rigid paddle for which we compute drag and lift effects. These effects alone are not enough to simulate the forces generated by an oscillating paddle. We use the average thrust model calculated by Plamondon and Nahon [20], and provide a set of parameters to tune the effects of paddle motion on the robot's body rotational motion [3].

Our simulator and physical robot share a common software API. As a result, it is possible to learn policies using reinforcement feedback from the simulator, following the same protocol that is used on the real robot. That is, we execute parameterized policies for fixed length episodes, collect experience and learn improved policies with PILCO. The control policies learned through this process share many properties to those learned on the real robot. Figure 1 shows one such policy, a 180 degree yaw turn, learned using reinforcement through interaction with our simulation environment.

D. Transfer of Simulated Learning Information

The results of learning based on our simulator also provide valuable information with the potential to benefit the real robot. Inspired by this intuition, we consider several *informed* learning approaches that transfer information between the simulator and the real robot. Each approach transfers one or both of: (a) the control policy learned in the simulator as a starting iterate for policy search on the real robot; and (b) episodic experience from the simulator in place of, or to augment the usual random control experience that is used to bootstrap the RL procedure.

We build a number of transfer techniques by making different choices in both steps (a) and (b). The potential choices in each step are enumerated here for use later to present our results. The bootstrap experience for learning can be produced with the following methods:

- (EXP-REAL-RAND) executes random commands on the real robot
- (EXP-SIM-RAND) executes random commands on in the simulated; (EXP-SIM-ALL) by collecting all experience data available in the simulator throughout a simulated learning process, which includes both the random commands and the commands of policies that are learned sequentially and converge towards lower loss
- (EXP-MIXED-ALL) combines random commands executed on the robot with the trace of simulated learning.

The initial policies iterate can be generated with the following methods:

- (POLI-RAND) is the default random initial policy based on no prior experience
- (POLI-SIM-FROM-SIM) is a policy learned on the simulator using purely on simulated data
- (POLI-SIM-FROM-REAL) is a policy learned on the simulator using bootstrapped from (EXP-REAL-RAND) data but proceeding with trials in the simulator, which we note features a mix of experience across domains

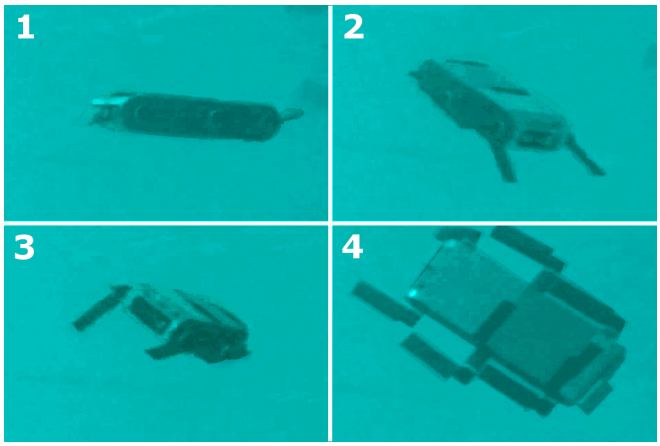


Fig. 5. The Aqua robot executes a policy learned for task 6 (180° yaw and 90° roll) during our experiments. Note that this motion is being executed using only two flippers for propulsion.

We will discuss two sample transfer techniques in full, to aid the reader with intuition. First, the technique pairing (EXP-REAL-RAND and POLI-RAND) is the *tabula rasa* learning baseline. It involves bootstrapping from random command data on the real robot and begins search from a random policy parameter. No information at all is drawn from the simulator. In contrast, the technique labeled (EXP-MIXED-ALL and POLI-FROM-SIM-REAL) is significantly different. It entails: collecting a small amount of robot experience with a random policy; using this data to bootstrap the process of learning a policy for the simulator until convergence; and finally bootstrapping with all experience, both simulated and real, plus the best policy learned in the simulator to perform learning on the real robot.

V. EXPERIMENTAL RESULTS

We continue by describing the results of learned underwater swimming performed by the Aqua hardward platform using our method. In all of the results displayed here, PLC commands were generated at a rate of 1 Hz, demonstrating the effectiveness of our periodic representation. We learned as many as 20 tasks in parallel using our cloud-computing parallel method.

A. Tabula Rasa

We attempted to learn all tasks on the real robot with as little human input as possible. For each task, a human specified only the loss function target. The initial policies used by the robot were produced by sampling RBF centers and targets at random, which result in highly ineffective initial swimming performance. In this scenario, it is entirely up to the learning algorithm to produce effective swimming policies.

Qualitatively, each of the seven tasks was learned effectively. Human observers could infer within one or two learning iterations which task was being attempted, and performance was visually similar to that of our previous human engineered controller within five iterations. This was quite noteworthy given the learner performed the tasks with only two flippers, while the previous methods had used all six.

Figure 4 displays a selection of quantitative learning results on for two representative tasks. For the first five trials, in grey, bootstrap experience is collected by swimming with a random policy. This produces large loss as the robot is far from the target state. The learning proceeds quickly achieves lower loss as the state trajectories converge to their targets with ever-increasing speed and stability. Note that task 1), the U-turn shown on the left, is a static angular target, while task 6), the corkscrew on the right, requires a constant angular rate. Both types of motions can be learned effectively.

B. Informed Learning

We have evaluated the *informed* learning techniques proposed in Sec. IV-D by implementing six approaches to share policies and/or experience from our simulator with the real robot. For each transfer method, we attempted to learn a sub-set of four swimming tasks: 1) U-turn, 3) belly-up, 4) corkscrew, and 6) knife-edge plus U-turn. We averaged learning performance over the tasks in order to compare the effect of each transfer approach.

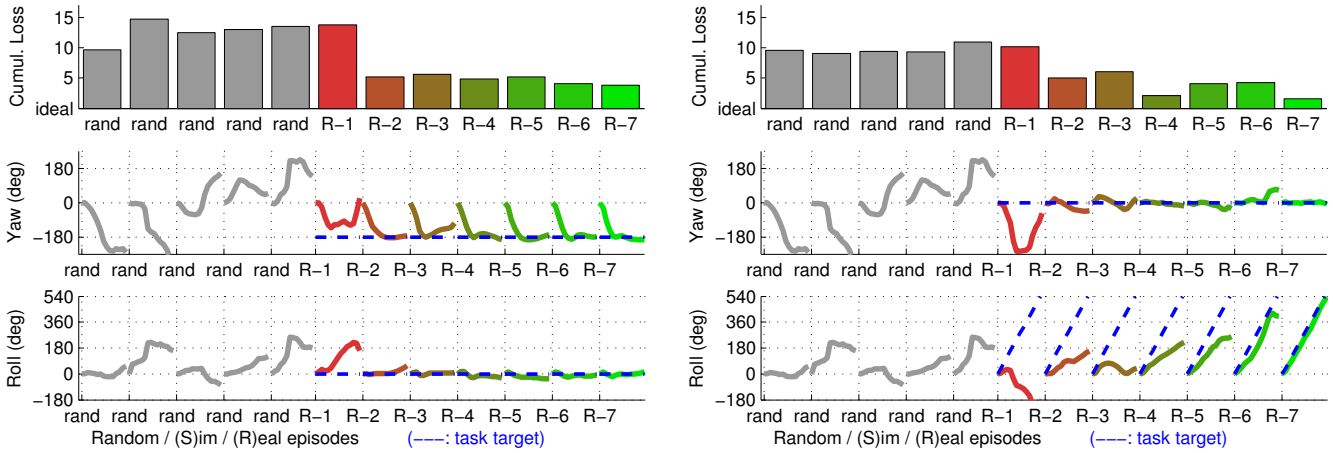
Qualitatively, our human operators reported several benefits resulting in transferring policies from the simulator to the real robot. The operators watched the execution of these seed policies before deploying the robot in the water. This allowed them to predict the initial motion of the real robot. The perceived safety and comfort of human participants is an interesting avenue for future study in this aspect.

Table I summarizes the results of each *informed* learning technique over the four tasks used in this analysis. There is a noteworthy negative correlation between task performance and the use of experience from the simulator to bootstrap learning. This is likely due to the differences in the dynamics models leading to less accurate GP predictions and state rollouts. We plan to investigate further calibration and more sophisticated transfer learning approaches that may allow simulated experience to be more fruitfully utilized by our method in the future.

VI. CONCLUSIONS

This paper has presented a method that allows a hexapod robot to learn to swim effectively based on its experience. We build upon the powerful PILCO [9] method, adapting it to swimming through use of radial basis function policies that generate periodic swimming commands at a coarse timescale. In the *tabula rasa* setting, many tasks were successfully learned bootstrapped from random exploration over controllers. The policies learned by our system utilize substantially different flipper motion than the previous hand-coded gait system that was available on our robot, unlocking the potential for significant adaptation to new tasks and environments.

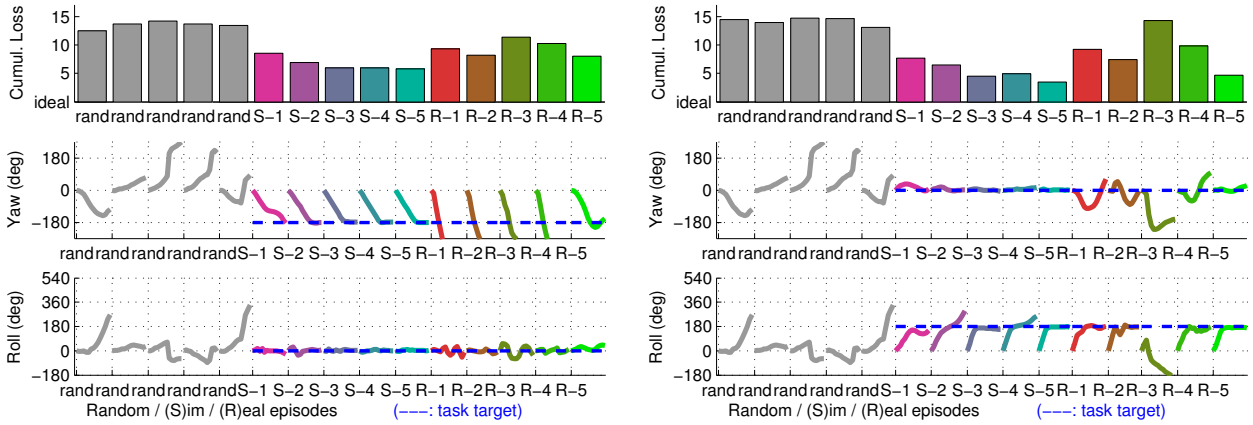
Through the development in this paper, we found the major shortcoming of PILCO for learning policies was the long learning run time. We have partially addressed this issue by interleaving learning of multiple tasks and using parallel



(a) U-turn task (180° yaw, 0° roll & pitch)

(b) corkscrew task (40°/s roll, 0° yaw & pitch)

Fig. 4. Example learning trial for (a) task 1) U-turn and (b) task 6) corkscrew tasks, learned *tabula rasa*. The target yaw and roll angles for each task are shown as dotted lines. The first five episodes use random control actions to generate bootstrap experience, which does not bring the robot near to its goal. Learning starts at the sixth iteration and the robot's state can be seen to converge to the target with increasing speed and stability as time progresses, which is reflected by reduced loss.



(a) Task 1 - U-turn task (180° yaw, 0° roll & pitch)

(b) Task 3 - belly-up (180° roll, 0° yaw & pitch)

Fig. 6. Example learning trials for (a) U-turn and (b) belly-up tasks, learned *tabula rasa*, with informed learning using the simulator with both the transfer of random and learned experience (SIM-ALL), as well as the learned policy in simulation (SIM-FROM-SIM).

Transfer Setup	Seed Experience	Initial Policy	First-iteration Loss	Min Loss	Mean Loss	Max Loss
1	REAL-RAND	RAND	7.84	5.11	8.22	8.70
2	SIM-RAND	RAND	14.28	10.29	12.42	14.50
3	REAL-RAND	SIM-FROM-SIM	13.61	10.26	12.88	14.42
4	REAL-RAND	SIM-FROM-REAL	11.03	8.78	11.35	13.53
5	SIM-ALL	SIM-FROM-SIM	13.72	11.35	12.62	13.72
6	MIXED-ALL	SIM-FROM-REAL	11.78	8.50	10.80	12.48

TABLE I

PERFORMANCE COMPARISON OF THE DIFFERENT TRANSFER TECHNIQUES, BASED ON THE CUMULATIVE LOSS PER LEARNING ITERATION (LOWER IS BETTER).

computation. However, in the future we plan to investigate further gains in efficiency which will allow more rapid adaptation to situations and the dynamic changes common in underwater settings.

In future work, we plan to investigate the chaining of multiple learned policy portions into longer trajectories in a fashion that is adaptive to changes in the local environment. We will also continue to explore methods that give human operators the ability to influence the learning process.

ACKNOWLEDGMENT

We would like to acknowledge the NSERC Canadian Field Robotics Network (NCFRN) for its funding support.

REFERENCES

[1] J. Sattar, G. Dudek, O. Chiu, I. Rekleitis, P. Giguère, A. Mills, N. Plamondon, C. Prahacs, Y. Girdhar, M. Nahon, and J.-P. Lobos, "Enabling autonomous capabilities in underwater robotics," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'08)*, 2008, pp. 3628–3634.

[2] C. Georgiades, "Simulation and control of an underwater hexapod robot," Master's thesis, McGill University, 2005.

[3] P. Giguere, C. Prahacs, and G. Dudek, "Characterization and modeling of rotational responses for an oscillating foil underwater robot," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'06)*, 2006, pp. 3000–3005.

[4] N. Plamondon, "Modeling and control of a biomimetic underwater vehicle," Ph.D. dissertation, McGill University, 2010.

[5] J. Weingarten, M. Buehler, R. Groff, and D. Koditschek, "Gait generation and optimization for legged robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'02)*, 2002.

[6] S. Chernova and M. Veloso, "An evolutionary approach to gait learning for four-legged robots," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, vol. 3, 2004, pp. 2562–2567.

[7] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'10)*, 2010, pp. 2397–2403.

[8] D. Meger, F. Shkurti, D. C. Poza, P. Giguère, and G. Dudek, "3D trajectory synthesis and control for a legged swimming robot," in *Proc. of the IEEE Int. Conf. on Robotics and Intelligent Systems (IROS'14)*, 2014.

[9] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. of the Int. Conf. on Machine Learning (ICML'11)*, pp. 465 – 472.

[10] D. Terzopoulos, X. Tu, and R. Grzeszczuk, "Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world," *Artificial Life*, vol. 1, no. 4, pp. 327–351, 1994.

[11] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'04)*, 2004, pp. 2619–2624.

[12] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, 2004, pp. 2849–2854.

[13] K. H. Low, C. Zhou, and Y. Zhong, "Gait planning for steady swimming control of biomimetic fish robots," *Advanced Robotics*, vol. 23, no. 7-8, pp. 805–829, 2009.

[14] A. Crespi and A. J. Ijspeert, "Online optimization of swimming and crawling in an amphibious snake robot," *IEEE Trans. on Robotics*, vol. 24, no. 1, pp. 75–87, 2008.

[15] A. Xu, A. Kalmbach, and G. Dudek, "Adaptive parameter exploration (APEX): Adaptation of robot autonomy from human participation," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'14)*, 2014, pp. 3315–3322.

[16] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Proc. of the 16th Int. Symposium on Robotics Research (ISRR'13)*, 2013.

[17] S. Barrett, M. E. Taylor, and P. Stone, "Transfer learning for reinforcement learning on a physical robot," in *9th Int. Conf. on Autonomous Agents and Multiagent Systems - Adaptive Learning Agents Workshop (AAMAS-ALA'09)*, 2010.

[18] P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proc. of the Int. Conf. on Machine Learning (ICML'06)*, 2006, pp. 1–8.

[19] M. Cutler, T. J. Walsh, and J. P. How, "Reinforcement learning with multi-fidelity simulators," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'14)*, 2014, pp. 3888–3895.

[20] N. Plamondon and M. Nahon, "Adaptive controller for a biomimetic underwater vehicle," *Journal of Unmanned Vehicle Systems*, vol. 01, no. 01, pp. 1–13, 2013.

[21] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'04)*, vol. 3, 2004, pp. 2149–2154.

[22] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and gazebo," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer Berlin Heidelberg, 2012, vol. 7628, pp. 400–411.