Learning Legged Swimming Gaits from Experience

David Meger[1], Juan Camilo Gamboa Higuera[1], Anqi Xu[1], Philippe Giguère[2] and Gregory Dudek[1]

***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 state-of-the-art policy search technique with a family of periodic low-level controls that are well suited for underwater propulsion. We demonstrate the practical efficacy of *tabula rasa* learning, that is, learning without the use of any prior knowledge, of policies for a six-legged swimmer to carry out a variety of acrobatic maneuvers in three dimensional space. We also demonstrate *informed* learning that relies on simulated experience from a realistic simulator. 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 a desired pose, or through an acrobatic motion such as a corkscrew. The success of our learning framework is assessed both in simulation and in the field using an underwater swimming robot.**

I. INTRODUCTION

In this paper, we study the task of learning swimming controllers for the Aqua family of hexapod amphibious robots [1], which employ six flexible flippers to generate thrust. 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 the *Karman street* flow pattern, which is 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 for handling similarly complex problem aspects [5], [6], [7], [8].

This work represents the first leg controller for Aqua to be based entirely on swimming experience from the vehicle. Previously, the vehicle was driven using expert-engineered swimming gaits [3] and motion controllers [9], [10]. Adaptive methods were only demonstrated in simulation [11]. 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. This includes dynamically altering the use of individual flippers between stationary dragging and active propulsion, yielding greater task efficiency. Various combinations of elementary motions have been observed in policies for the same high-level task, representing the flexibility of our approach. For

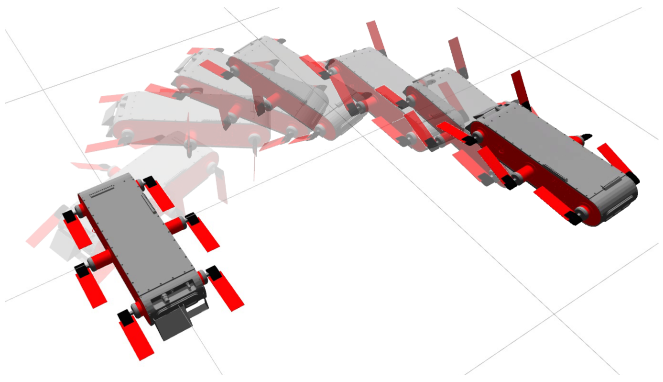


Fig. 1. A learned policy for a 6 flipper swimming task is executed within the Gazebo simulation environment used to validate our approach. The task learned was a 180° yaw (U-turn).

example, the task "U-turn", has been executed by remaining flat, by banking moderately, or by rolling completely sideways. Furthermore, even when using a highly under-actuated 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.

Our gait learning approach is based on a policy search technique named PILCO (Probabilistic Inference for Learning Control) [12], 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 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, (i.e., 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.

Following a brief survey of related work, Section III describes our reinforcement learning (RL) solution for *tabula rasa* learning of swimming controllers. Section 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 Section V.

[1]Mobile Robotics Lab, School of Computer Science, McGill University {dmeger,gamboa,anqixu,dudek}@cim.mcgill.ca
[2]Département d'informatique et génie logiciel, Faculté de science et génie, Université Laval philippe.giguere@ift.ulaval.ca

II. BACKGROUND

A. Gait Learning

Gait optimization and discovery is a long-established and well-studied research area. Works include the use of simulated muscle actuators for legged walking, crawling and even swimming, albeit for idealized entities [13]. Several authors have considered gait learning for physical robots operating on land [6], [7], [14], [15], [8]. Relatively few authors have considered swimming gaits for real robots, with [16] being an exception, which targeted a robot with simpler actuation than our own. 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 [17]. A major challenge to learning in the marine domain is the lack of reliable ground truth due to the degraded RF and visual sensing conditions [18].

In the experiments below, we evaluate the use of a simulator to bootstrap learning on a real robot. This choice is inspired by the recent success of demonstrations or direct participation as effective aids for learning practical robotic tasks [19], [20]. Additionally, numerous authors [21], [22], [23] have demonstrated benefits from transfer of information both forward and backwards between the learning procedures of the simulated and real robots. The use of an inaccurate analytical model to facilitate policy search approaches was notably analyzed in [22].

B. PILCO

The PILCO method [12] has been capable of learning controllers for systems ranging from 1-D 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 summarize its algorithmic 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, $\sum_{t=0}^{T} \mathbb{E}\left[L(\mathbf{x}_t)\right]$. $L(\mathbf{x})$ is a loss function provided by the system designer to specify the target task. Policy search requires parameterized policies $\mathbf{u} = \pi(\mathbf{x}, \theta)$, capable of generating commands $\mathbf{u}$ based on state $\mathbf{x}$. 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 $\mathbf{\Delta}_t = (\mathbf{x}_{t+1} - \mathbf{x}_t)$ based on previous state and control action: $\mathbf{\Delta}_t \sim GP(\mathbf{x}_t, \mathbf{u}_t)$.

PILCO estimates state (equivalently: loss) trajectories using an analytical approximation of the state distribution and its derivatives after rolling out a policy for T time steps. This allows the use of gradient-based search techniques over θ to evaluate 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 use of an efficient control space encoding periodic behaviors, rather than direct position control of each of our robot's six 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. We disable all previously developed sensor-based control and coordination modules, only providing the lowest-level motor and sensing drivers that allow our learning module to operate the vehicle. In each task episode, the robot executes a control policy, which generates a *periodic leg command* (PLC) based on the sensory state at every time step. Over multiple 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 is denoted by $\mathbf{x}$, at each time step. Notably, this is also the space over which the user is able to define the loss (i.e., negative reward) function, $L(\mathbf{x})$. We have investigated two families of loss functions in order to build up a dynamic set of motions for our robot: *static-angle, static-depth* tasks that ask the robot to achieve and stably maintain a target orientation; and *dynamic-angle, static-depth* tasks that require the robot to regulate its angular rate, such as a corkscrew. In this work we do not consider positional feedback, although this data can be readily incorporated in our solution when the swimming robot is equipped with a 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 [24] or other modeling techniques. These motions include oscillation, spinning, and braking by statically opposing water flow. Inspired by previous work in learning control primitives [25], we expose a naturally useful set of low-level controls to the learning mechanism.

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

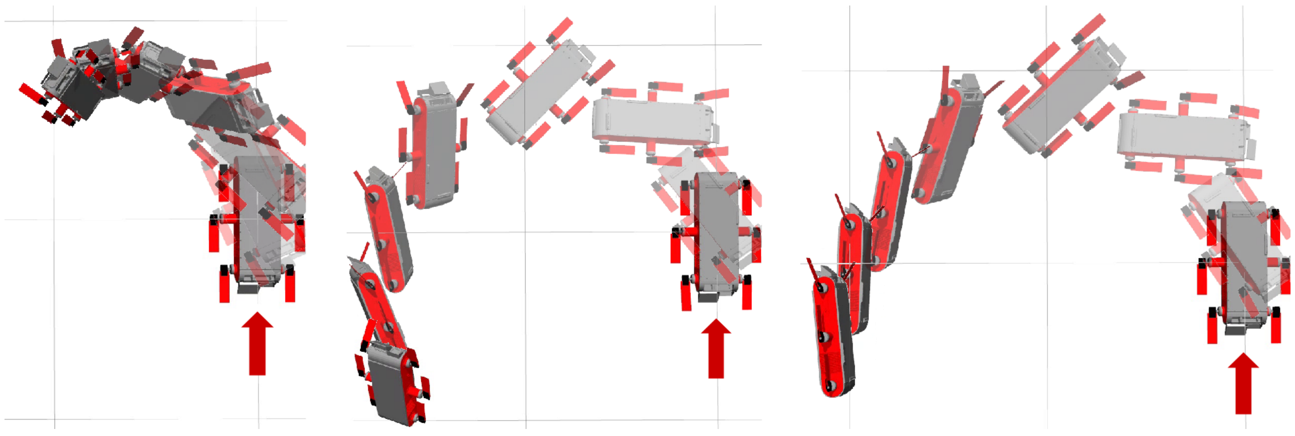

(a) Episode 1 (b) Episode 5 (c) Episode 10

Fig. 2. Three learning iterations on the simulator for task 6: *knife-edge plus U-turn* (180° yaw and 90° roll targets). 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. The robot starts each trial from the right-most pose in each panel and proceeds in the direction indicated by the red arrow.

$\mathbf{u}_i$: $\phi_i = \text{PLC}(t, \mathbf{u_i})$. The PLC function is a sinusoid of the form:

$$\text{PLC}(t, \mathbf{u}_i) = A_i \cdot sin(2\pi f_i t + \varphi_i) + b_i \tag{1}$$

where each command parameter vector, $\mathbf{u}_i$, contains amplitude, A_i, frequency, f_i, phase, φ_i, and angular offset, b_i. 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 $\mathbf{u} = [\mathbf{u}_1, ..., \mathbf{u}_6]$ vector will keep the legs continuously in motion, until the next command is received. This allows policies to be efficiently learned at a coarse time scale, saving significant computational cost.

C. RBF Policies

We require a policy family that is able to map robot states into PLC actions for each time step, and which is parameterized to allow PILCO to perform loss optimization: $\mathbf{u} = \pi(\mathbf{x}, \theta)$. We employ Radial Basis Functions (RBF) as policies, which have the form:

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

$$\gamma_i(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_i)^T \mathbf{\Lambda}^{-1}(\mathbf{x} - \mathbf{c}_i)\right) \tag{3}$$

The parameters of each RBF component are its center, c_i, in the sensor state space, and a corresponding weight, t_i. In our work, this RBF policy is obtained as the mean prediction of a Gaussian Process regression, as in [26].

D. Loss Function

A human designer must create a loss function, $L(x)$, for characterizing a desired behavioral task to be learned. This function can be defined arbitrarily in terms of the state-space, thus allowing for a wide range of target tasks. To ensure smoothness, we apply a unity minus Gaussian function with width σ centered at the specified goal. The cost function width is an important factor in ensuring learning convergence, and is set manually using design heuristics.

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 thus we have implemented a cloud-based parallelization method in order to effectively learn multiple tasks within a reasonable time-frame. We investigate *tabula rasa* learning, (i.e., without relying on prior system knowledge), as well as *informed* learning that utilizes information transferred from a simulator.

A. Sample Tasks

A broad range of gait learning tasks were explored during our system evaluation, as can be seen 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 six fixed-depth tasks:

1) *U-turn*: flat 180° yaw
2) *Knife-edge*: straight-ahead 90° roll
3) *Belly-up*: straight-ahead 180° roll
4) *Fast corkscrew*: forward clockwise rapid roll change
5) *Slow corkscrew*: forward anti-clockwise slow roll change
6) *Knife-edge plus U-turn*: 180° yaw and 90° roll

We have executed learning for each of these tasks in simulation using the vehicle's full 6 flipper propulsion system – a task consisting of 13 state and 12 control dimensions. One

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

sample learned trajectory can be viewed in the accompanying video. To increase the challenge for our data-driven method and reduce the state dimensions to allow for feasible experimentation on real hardware, we have restricted the robot to swimming with only its back two flippers for the quantitative analysis shown below. This highly under-actuated scenario requires precise control and coordination of the two available control surfaces. Each two flipper task is specified in a 7-D state space (3 angles, 3 angular rates, and depth). The policy must produce four-dimensional control outputs consisting of amplitude and offset for 2 legs. The phase and frequency are not utilized. Tasks are executed for a fixed horizon of 15 seconds.

B. Parallelized Off-board Policy Search

Learning with PILCO is many times slower than real-time for the swimming tasks that we consider on a typical desktop workstation. This presents a practical problem, since experiment time on the physical robot is expensive in terms of power consumption and human labor. To minimize the robot's idle time, and make best use of 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, the experience data is transferred off-site and a remote learning job is instantiated, all the while the robot moves onto learning a subsequent task. When the remote learning job is completed, the resulting policy is transmitted back to the robot and the next episode for that learning task is entered into a queue. We have learned up to 20 tasks in parallel using this framework, allowing our robot to run nearly continuously, despite the long learning cycles for each task.

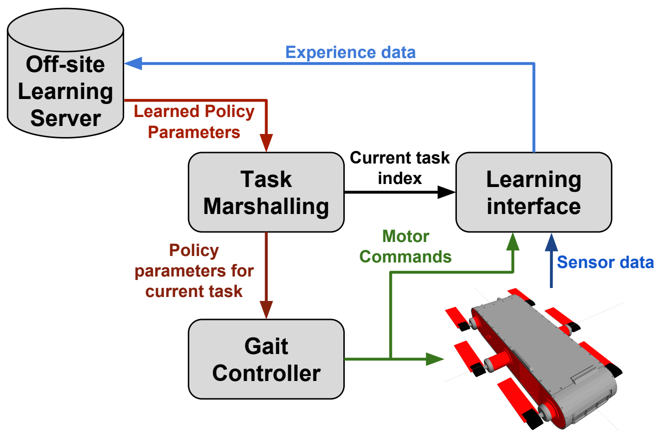


Fig. 3. A visual description of the basic elements in our parallelized off-board policy search system.

C. Swimming Simulator

We have developed an underwater simulation environment for the Aqua robot. This implementation includes a set of plug-ins for the Gazebo simulator [27]. 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 [28], our plug-ins 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 linear and rotational drag, added mass, buoyancy and gravity, following the description of these effects in [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 due to their non-linear dependence on turbulent flow patterns. We use the average thrust model from [24], and provide a set of parameters to tune the effects of paddle motion on the robot's body rotation [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° yaw turn, learned using reinforcement through interaction with our simulation environment.

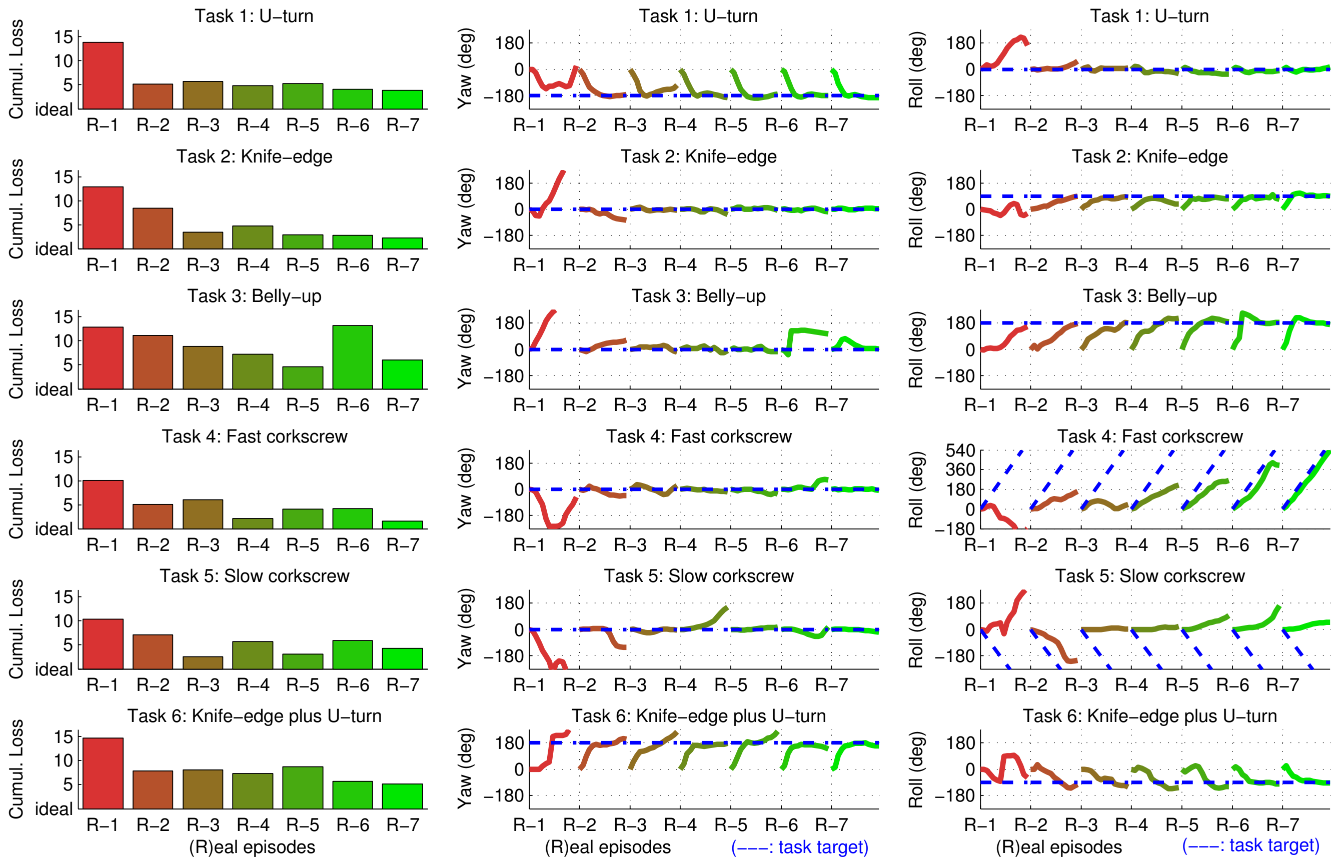
D. Transfer of Simulated Learning Information

The results of learning based on our simulator 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 an initial seed point 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 analyze several transfer techniques that use different choices in 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 the simulator
- (EXP-SIM-ALL) collects all experience data available in the simulator throughout a simulated learning process, which includes both the random commands and the commands of learned policies
- (EXP-MIXED-ALL) combines random commands executed on the robot with the trace of simulated learning

The initial policy on the real robot is generated using one of the following methods:



(a) Cumulative Loss (b) Yaw Angle (c) Roll Angle

Fig. 4. *Tabula rasa* learning trials for all 6 fixed-depth tasks. The target yaw and roll angles for each task are shown as dotted lines. For each task, five initial episodes using random control actions (not shown) generated bootstrap experience, which does not bring the robot near to its goal. Learning then starts 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. Task 5 (*slow corkscrew*) successfully learns to regulate the target yaw angle, but failed to achieve target roll rate.

- (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 simulated data
- (POLI-SIM-FROM-REAL) is the policy resulting from (EXP-MIXED-ALL)

We now elaborate on two sample transfer techniques, to provide further intuition. Firstly, 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 randomized policy setting. This technique does not make use of the simulator at all.

In contrast, the technique labeled (EXP-MIXED-ALL and POLI-SIM-FROM-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 till convergence; and finally bootstrapping with all experience, both simulated and real, coupled with the best policy learned in the simulator to perform learning on the real robot.

V. EXPERIMENTAL RESULTS

This section describes the results of learned underwater swimming behaviors performed by the Aqua hardware platform using our method. In all of the displayed results, PLC commands were generated at a coarse rate of 1 Hz, demonstrating the efficacy of our periodic representation. During experimental sessions, as many as 20 tasks were learned simultaneously through a cloud-computing implementation of our parallelized off-board learning architecture.

A. *Tabula Rasa*

We carried out *tabula rasa* learning of all 6 tasks on the real robot, with as little human input as possible. For each task, the human designer specified solely the loss function target. The initial policies used by the robot were produced by sampling RBF centers and targets at random, which resulted in highly ineffective initial swimming performance. The quality of the swimming motions in subsequent trials is entirely dependent on the success of the learning algorithm.

Figure 4 depicts quantitative learning results for each of the six tasks. Each sequence was bootstrapped with five initial trials of random motion, which are not displayed. The

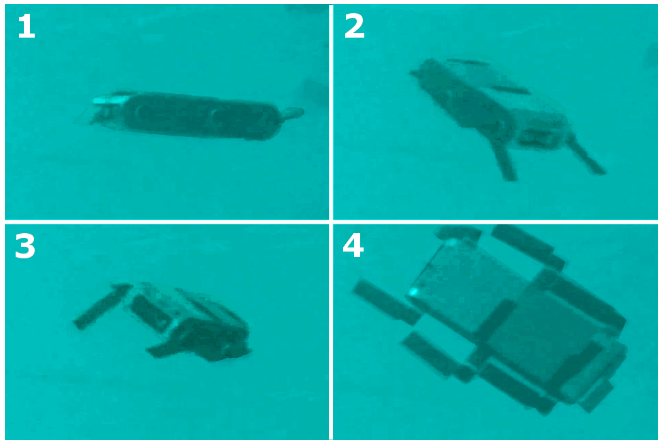


Fig. 5. The Aqua robot executes a policy learned for task 6 – *Knife-edge plus U-turn* (180° yaw and 90° roll) during our experiments. Note that this motion is being executed using only two flippers for propulsion.

learning process then swiftly achieved low loss as state trajectories converged to their targets with ever-increasing speed and stability. Notably, tasks 1, 2, 3, and 6 employed static angular targets, whereas tasks 4 and 5 (i.e., corkscrew motions) required constant angular rates. Our results demonstrated that both types of motions could be learned effectively.

Five out of the six tasks were learned with quality that matched our previous human engineered controller within seven iterations. This was quite noteworthy given that Aqua learned to perform challenging swimming motions using only two flippers (e.g., Figure 5), whereas previous control methods required actuating all six flippers. The slow corkscrew task did not converge to satisfactory performance within our experimental session, which may be due to our experimental protocol introducing an unrecoverable error in the dynamics model in one of the early learning iterations for this task.

B. *Informed Learning*

We have evaluated the *informed* learning techniques proposed in Section 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 four of our six swimming tasks: 1) U-turn, 3) belly-up, 4) corkscrew, and 6) knife-edge plus U-turn. Learning performance were averaged across tasks to compare the effect of each transfer technique.

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 near future.

VI. Conclusions

This paper presented a method that allows a hexapod robot to learn to swim effectively based on its experience. We built upon the powerful PILCO [12] method, and adapted it for swimming tasks through the use of radial basis function policies that generate periodic swimming commands at a coarse timescale. In a *tabula rasa* manner, many tasks were successfully learned by bootstrapping from random exploration over controllers. The policies learned by our system utilized substantially different flipper motion compared to the hand-engineered controller used previously on our robot. This success unlocks the potential for significant adaptation to new tasks, and enables the robot to recover gracefully from hardware failures.

Through the development in this work, we repeatedly encountered a major shortcoming of PILCO, namely its long learning run times. We have partially addressed this issue by interleaving learning of multiple tasks using parallel computation. We plan to further investigate gains in efficiency that 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 Int. Robots and Systems (IROS)*, 2008.

[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 IEEE/RSJ Int. Conf. on Int. Robots and Sys. (IROS)*, 2006.

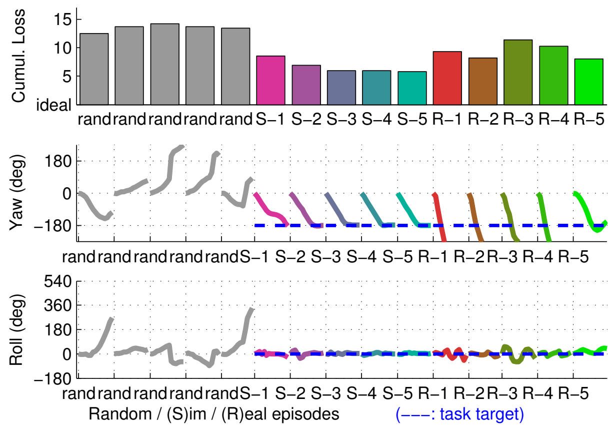
[4] N. Plamondon, “Modeling and control of a biomimetic underwater vehicle,” Ph.D. dissertation, McGill University, 2010.

[5] R. Calandra, N. Gopalan, A. Seyfarth, J. Peters, and M. Deisenroth, “Bayesian gait optimization for bipedal locomotion,” in *Learning and Intelligent OptimizatioN (LION8)*, 2014.

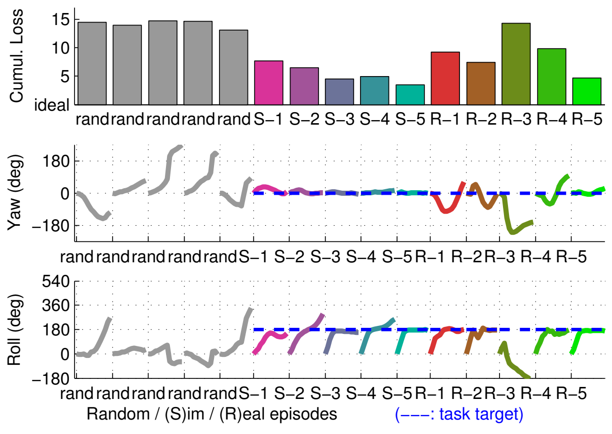
[6] J. Weingarten, M. Buehler, R. Groff, and D. Koditschek, “Gait generation and optimization for legged robots,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002.

[7] 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)*, 2004.

[8] 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)*, 2010.



(a) Task 1: *U-turn* (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, via 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 DIFFERENT TRANSFER TECHNIQUES, BASED ON CUMULATIVE LOSS PER LEARNING ITERATION (LOWER IS BETTER).

[9] P. Giguere, Y. Girdhar, and G. Dudek, “Wide-speed autopilot system for a swimming hexapod robot,” in *Proc. of the Canadian Conf. on Computer and Robot Vision (CRV)*, 2013.

[10] 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 Int. Robots and Systems (IROS)*, 2014.

[11] A. German and M. Jenkin, “Gait synthesis for legged underwater vehicles,” in *Proc. of the Int. Conf. on Autonomic and Autonomous Systems (ICAS)*, 2009.

[12] 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)*, 2011.

[13] 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, 1994.

[14] 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)*, 2004.

[15] 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 Int. Robots and Systems (IROS)*, 2004.

[16] 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, 2009.

[17] 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, 2008.

[18] L. A. Torres-Méndez and G. Dudek, “Color correction of underwater images for aquatic robot inspection,” in *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2005.

[19] 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 Auto. (ICRA)*, 2014.

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

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

[22] P. Abbeel, M. Quigley, and A. Y. Ng, “Using inaccurate models in reinforcement learning,” in *Proc. of the Int. Conf. on Machine Learning (ICML)*, 2006.

[23] 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)*, 2014.

[24] N. Plamondon and M. Nahon, “Adaptive controller for a biomimetic underwater vehicle,” *Journal of Unmanned Vehicle Systems*, 2013.

[25] S. Schaal, “Dynamic movement primitives-a framework for motor control in humans and humanoid robotics,” in *Adaptive Motion of Animals and Machines*. Springer, 2006.

[26] M. Deisenroth, D. Fox, and C. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[27] 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)*, vol. 3, 2004.

[28] 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, 2012, vol. 7628.