Proceedings of the 2024 IEEE
International Conference on Robotics and Biomimetics
December 10-14, 2024, Bangkok, Thailand

Fine Tuning Swimming Locomotion Learned from Mosquito Larvae

Pranav Rajbhandari[1] and Karthick Dhileep[2] and Sridhar Ravi[3] and Donald Sofge[4]

2024 IEEE International Conference on Robotics and Biomimetics (ROBIO) | 979-8-3315-0964-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ROBIO64047.2024.10907546

***Abstract*— In prior research, we analyzed the backwards swimming motion of mosquito larvae, and created a parametrized approximation in a Computational Fluid Dynamics simulation. Since the parameterized swimming motion is replicated from observed larvae, it is not necessarily the most efficient locomotion. In this project, we further optimize this swimming locomotion for the simulated platform, using Reinforcement Learning to guide local parameter updates. Since the majority of the computation cost arises from the Computational Fluid Dynamics model, we additionally train a deep neural network to replicate the forces acting on the swimmer model. We find that this method is effective at performing local search to improve the parameterized swimming locomotion.**

I. Introduction/Related Work

Bio-inspired robot designs are appealing as an alternative to traditional robots, as they mimic organisms that have adapted to a particular environment. In the setting of autonomous underwater vehicles, bio-inspired robots have been shown to have advantages such as increased energy efficiency [17] and reduced swimming noise [4]. However, these systems are often much more complex, and require sophisticated algorithms to control [20]. Thus, using learning algorithms to optimize the swimming locomotion of bio-inspired robotics is useful to develop more efficient movement for robotic swimmers. Additionally, this can lead to better understanding of the real organisms that these systems are based on [6].

A. Locomotion of Mosquito Larvae

In previous research, we parameterize the swimming motion of mosquito larvae and successfully replicate it inside a Computational Fluid Dynamics (CFD) simulator [3]. We model the swimmer as a 2D boundary and use the immersed boundary lattice Boltzmann method (IB-LBM) [10] to calculate forces and resulting trajectory of a swimming locomotion.

For the parametrization, we discretize the swimmer into 200 line segments and estimate the angle θ between adjacent segments. This angle varies with time $t \in \mathbb{R}$ as well as position along the body of the swimmer $s \in [0,1]$. We found in [3] that $\theta(s,t)$ was well represented by Equation 1, using an amplitude function $A(s)$, a frequency ω, and phase shift function $\phi(s)$.

[1]Pranav Rajbhandari is the corresponding author and with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, prajbhan@alumni.cmu.edu. They completed this work under NREIP at Naval Research Laboratory, Washington D.C., USA.

[2]Karthick Dhileep is with the School of Engineering and Technology, University of New South Wales, Canberra, Australia.

[3]Sridhar Ravi is with the School of Engineering and Technology, University of New South Wales, Canberra, Australia.

[4]Donald Sofge is with the Naval Research Laboratory, Washington D.C., USA, donald.a.sofge.civ@us.navy.mil.

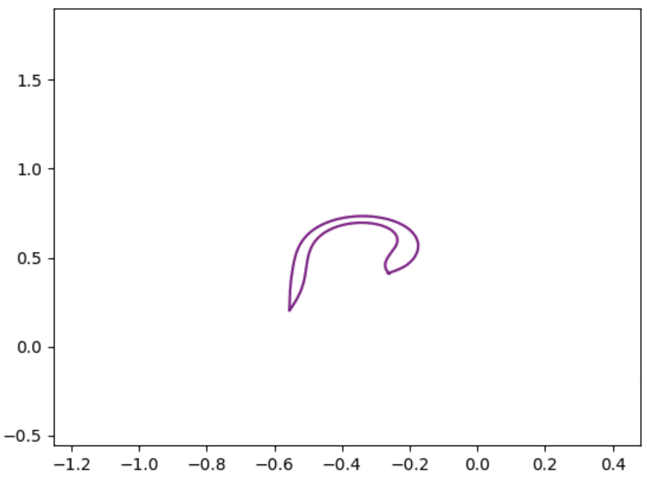


Fig. 1. Model of 2D swimmer in CFD

$$\theta(s,t) = A(s) \cdot \sin(\omega t + \phi(s)) \tag{1}$$

We approximate A and ϕ as polynomials with respect to s of degrees 5 and 4 respectively. In addition to ω, this results in a 12 dimensional parameter space. Explicitly, we may rewrite Equation 1 using a parameter vector $\mathbf{p} \in \mathbb{R}^{12}$:

$$\theta_{\mathbf{p}}(s,t) = \left(\sum_{i=0}^{5} s^i \mathbf{p}_{i+1}\right) \cdot \sin\left(\mathbf{p}_{12} t + \sum_{i=0}^{4} s^i \mathbf{p}_{i+7}\right) \tag{2}$$

We obtain initial parameters in [3] by estimating the motion of live mosquito larvae.

B. Local Search/Hill Climbing

The hill climbing algorithm is a well-known local search method that repeatedly updates a solution by sampling from a local neighborhood [15]. In continuous search spaces, this can be approximated by fixing a step size δ and searching around a solution by taking a δ step in every dimension. This estimates a gradient of the objective with respect to the parameter space, and can be calculated in $O(d)$ for d the number of dimensions.

We may apply this to optimizing the parameters of an initial swimming locomotion. We set our objective to displacement in some set time, and evaluate a solution through a simulation. With this method, a single update (assuming we take a full gradient estimation) will require $O(d)$ simulations.

This is a reasonable approach if we only utilize our simulation for evaluating a potential solution. However, by

Authorized licensed use limited to: University of New South Wales. Downloaded on August 01,2025 at 01:50:21 UTC from IEEE Xplore. Restrictions apply.

making adjustments to the swimming policy mid-episode, we can better estimate what actions increase our objective. To make these adjustments, we utilize a Reinforcement Learning (RL) algorithm.

C. Baseline Guided Policy Search

Hu and Dear explore a similar problem of training an articulated robotic swimmer through RL [9]. They introduce Baseline Guided Policy Search (BGPS), an RL algorithm that allows an agent to add small adjustments to an existing baseline policy. In their research, they use this method to optimize swimming motion in robotic swimmers composed of three segments.

We utilize this technique to make adjustments mid-simulation to a swimming locomotion. We will then learn parameters that best approximate this adjusted policy, updating the baseline.

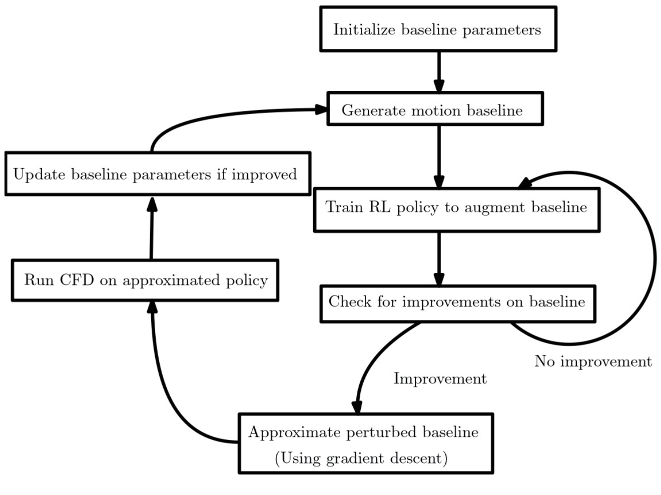
Since BGPS is restricted to making relatively small updates to the swimming motion, we expect that the update will be relatively small when projected to parameter space. Thus, this will behave similarly to a local search algorithm in the swimming parameters. The main distinction is the number of samples an update requires. While a standard local search would need to sample simulations of lots of neighboring solutions to make a gradient-based update, we expect an RL algorithm to find an improving update within a simulation or two by learning kinematic information throughout each episode.

II. Methods

A. Simulated Mosquito Swimmer

In previous research, we create a simulated mosquito larvae inside a CFD simulator. The setup is able to replicate the dynamics of real mosquito larvae from the copied swimming locomotion [3]. We utilize this CFD model to fine tune the locomotion of the simulated swimmer.

B. RL-Guided Parameter Update



Initialize baseline parameters
Generate motion baseline
Update baseline parameters if improved
Train RL policy to augment baseline
Run CFD on approximated policy
Check for improvements on baseline
No improvement
Improvement
Approximate perturbed baseline
(Using gradient descent)

Fig. 2. Local parameter search using BGPS algorithm

We implement a RL environment utilizing a CFD simulation. We use this environment to optimize a set of parameters, as in Figure 2. We first repeatedly run the BGPS algorithm, searching for an augmentation that outperforms the baseline policy. Once this policy is found, we approximate parameters to replicate the augmented policy. Finally, we test using CFD if these parameters are truly an improvement, and update the baseline accordingly.

To choose parameters that replicate an augmented policy, we sample angles $\theta^*(s,t)$ that the augmented policy θ^* predicts for positions s and times t in an episode. We consider values of t within one period of the swimming motion, and values of s that align with each joint of the swimmer. We then find parameters $\mathbf{p}$ such that $\theta_{\mathbf{p}}(s,t)$ from Equation 2 approximates $\theta^*(s,t)$ for our sampled (s,t). Since $\theta_{\mathbf{p}}$ is differentiable with respect to $\mathbf{p}$, we do this through gradient descent, minimizing the Mean Squared Error loss (Equation 3). We initialize our search with the baseline parameters, since $\theta^*(s,t)$ arises from small adjustments to this.

$$\mathcal{L}(\mathbf{p}) = \mathbb{E}_{s,t}\left[\left(\theta^*(s,t) - \theta_{\mathbf{p}}(s,t)\right)^2\right] \tag{3}$$

C. CFD Clone

Deep learning has become a powerful tool for approximating fluid dynamics calculations in the past decade [11], [12]. Approaches can be roughly split into *physics-driven* models, which approximate the underlying fluid dynamics equations, and *data-driven* models, which learn to replicate flow solutions without explicitly being trained on the underlying physics. Various model architectures have been used, including Convolutional Neural Networks [7], [18] and Recurrent Neural Networks [1], [2].

We create a data-driven model that predicts the forces acting on a simulated swimmer based on the movements of its outline. We experiment with various network architectures, and will use our best performing model to replicate CFD simulations.

To create the training data, we evaluate the CFD simulation on a sweep of parameterized swimming motions. For the model loss, we use the sum of mean squared error loss and cosine similarity loss to ensure the forces are correctly oriented.

1) Network Input: We allow the network to observe the outline of the swimmer at each timestep, centered at the swimmer's center-of-mass. This is a set of 400 sampled points on the swimmer surface. We use this as our network input since it is identical to the CFD model input. For our feed-forward network, we allow the network to observe the past three timesteps so it may obtain kinematic information about the swimmer.

2) Network Output: The network output is the surface forces on each of the 400 surface points. We use this as our network output since it is the output of the CFD, and is sufficient to calculate the movement of the swimmer.

Authorized licensed use limited to: University of New South Wales. Downloaded on August 01,2025 at 01:50:21 UTC from IEEE Xplore. Restrictions apply.

3) CFD calculation: We use the trained model to create a CFD clone by calculating surface forces at every timestep and applying kinematic equations, similar to the calculations in [21].

III. Experiments

A. CFD Clone

We evaluate the Recurrent Neural Network (RNN) [14] and Long Short-Term Memory (LSTM) [8] sequence-to-sequence architectures. We also evaluate a residual network to compare with non-sequential methods.

We hypothesize that sequential models are better suited to handle the estimation of forces on our swimmer, as they can obtain information from the full history of the swimmer, allowing them to keep track of kinematic information and to be more robust to noise.

1) Network Architecture: In addition to varying the model used, we also evaluate different network sizes in their ability to reduce the objective function. We take our best performing model and vary the depth of the architecture from one layer to eight layers. In our final CFD clone, we use the simplest network that performs comparably well.

B. Baseline Guided Policy Search

We implement BGPS to make adjustments to a baseline swimming policy. As in Figure 2, we alternate between using BGPS to improve the baseline and fitting parameters to the augmented policy. We use the stable_baselines3 [13] implementation of Proximal Policy Optimization (PPO), a standard on-policy RL algorithm [16].

1) Observation Space: Guided by Hu and Dear's work [9], we allow the agent to observe a few angles on the its midline, its heading angle, its position and velocity, and the simulation time (encoded by applying sine and cosine at various frequencies [5], [19]).

2) Action Space: The action space available to the agent is a list of angles corresponding to joints on its midline. The actions from the RL agent are used to augment a baseline policy.

3) Rewards: We observe the normalized total displacement of the baseline policy's movement. At each timestep, we give the RL agent rewards equivalent to the displacement in the direction of the baseline displacement vector. We do this to ensure that the sum of rewards in an episode is the swimmer's overall displacement in the same direction as the baseline policy.

IV. Results

A. CFD Clone

We inspect the test loss of the LSTM, RNN, and residual network models during training. We find that LSTM performed the best, followed by the RNN, and then the residual network (Figure 3). We observe that both the sequential models outperformed the residual network, which supports that sequential architectures are better suited to estimate kinematic information throughout an episode.

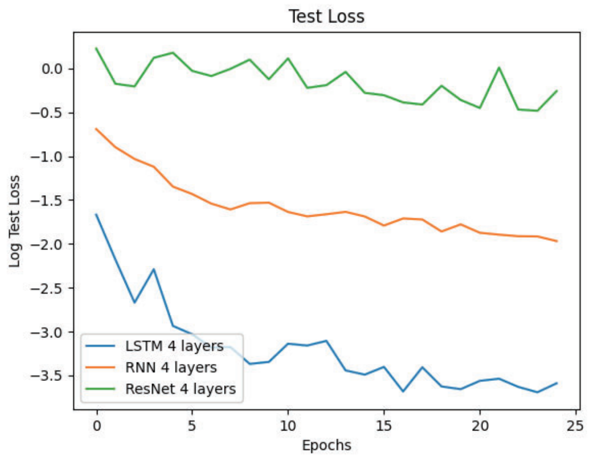


Fig. 3. Comparison of log test losses of LSTM, RNN, and Residual Network (ResNet) models

Since the LSTM architecture outperformed the others, we proceed to evaluate the performance of various LSTM network sizes.

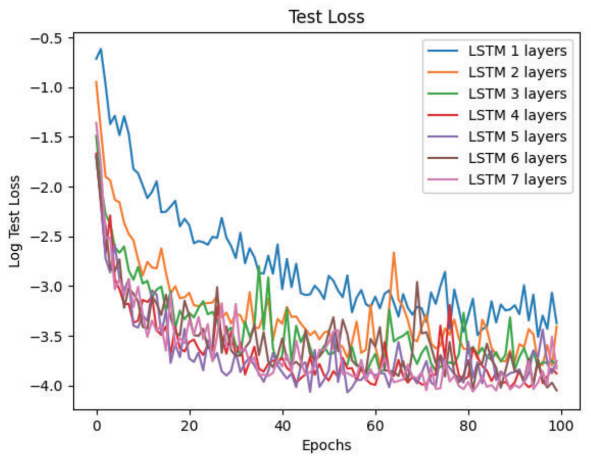


Fig. 4. Comparison of log test losses of various LSTM network depths

1) Network Architecture: We test and compare LSTM networks ranging from one to seven layers in depth. We notice that at depths of 1 and 2 the networks perform worse with respect to their test loss (Figure 4). The networks at higher depths all perform similarly. Since a depth of 3 is the most shallow network that performed well, we use this depth in our CFD clone.

B. BGPS

We use the resulting CFD clone to optimize our swimming locomotion from the initial choice of parameters. In each episode of training, we record the absolute value of the total displacement.

From Figure 5, we find that the BGPS algorithm is successful in gradually optimizing the movement of the simulated swimmer.

Authorized licensed use limited to: University of New South Wales. Downloaded on August 01,2025 at 01:50:21 UTC from IEEE Xplore. Restrictions apply.

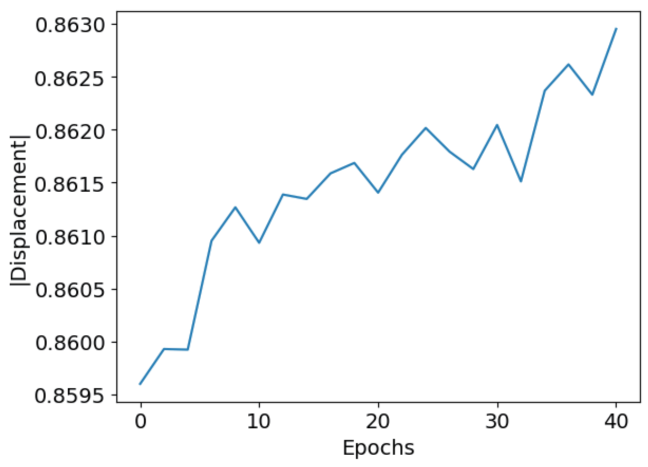


Fig. 5. Result of BGPS on swimmer displacement per episode

However, the scale of the improvement is small in comparison to the size of the displacement. This could be a result of BGPS only augmenting the policy on small scales.

V. Conclusion

In this study, we fine tune a learned parameterized swimming locomotion for a specific platform. We use local search to gradually update the parameters towards more optimal neighbors. To increase efficiency, we use RL to learn kinematic information about the swimming locomotion, guiding the local search.

We additionally approximate the learning environment with a CFD clone, learned through a deep neural network. We utilize this CFD clone to efficiently conduct model-based RL to improve the baseline policy. Overall, we take advantage of kinematic nature of our optimization problem to improve the speed of local search.

We find that these methods are successful in improving the parameterized swimming locomotion through local search. However, we find that the scale of the improvements are small.

In future research, we plan to vary the amount that BGPS can augment the policy to obtain more drastic differences. We also plan to use this method to optimize locomotion on a physical robotic swimmer.

References

[1] Sandeep Bukka, Allan Magee, and Rajeev Jaiman. Deep convolutional recurrent autoencoders for flow field prediction. volume 8: CFD and FSI of *International Conference on Offshore Mechanics and Arctic Engineering*, page V008T08A005, 2020.

[2] Sandeep Bukka, Allan Magee, Rajeev Jaiman, J. Liu, W. Xu, A. Choudhary, and A. A. Hussain. Reduced Order Model for Unsteady Fluid Flows via Recurrent Neural Networks. volume 2: CFD and FSI of *International Conference on Offshore Mechanics and Arctic Engineering*, page V002T08A007, 2019.

[3] Karthick Dhileep, Qiuxiang Huang, Fangbao Tian, John Young, Joseph C.S. Lai, Donald Sofge, and Sridhar Ravi. Investigation of bio-inspired tail-first swimming using numerical and robotic models. In *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1–6, 2023.

[4] Frank E. Fish. Advantages of aquatic animals as models for bio-inspired drones over present AUV technology. *Bioinspiration & Biomimetics*, 15(2):025001, 2020.

[5] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org, 2017.

[6] Nick Gravish and George V. Lauder. Robotics-inspired biology. *Journal of Experimental Biology*, 221(7):jeb138438, 2018.

[7] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional Neural Networks for Steady Flow Approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 481–490, New York, NY, USA, 2016. Association for Computing Machinery.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] Jiaheng Hu and Tony Dear. Guided Deep Reinforcement Learning for Articulated Swimming Robots, 2023.

[10] Qiuxiang Huang, Zhengliang Liu, Li Wang, Sridhar Ravi, John Young, Joseph Lai, and Fang-Bao Tian. Streamline penetration, velocity error, and consequences of the feedback immersed boundary method. *Physics of Fluids*, 34(9), 2022.

[11] J. Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

[12] Mario Lino, Stathi Fotiadis, Anil A. Bharath, and Chris D. Cantwell. Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479(2275):20230058, 2023.

[13] Antonin Raffin et al. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[14] David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*. MIT Press, 1987.

[15] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017.

[17] M. Sfakiotakis, D.M. Lane, and J.B.C. Davies. Review of fish swimming modes for aquatic locomotion. *IEEE Journal of Oceanic Engineering*, 24(2):237–252, 1999.

[18] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows. *AIAA Journal*, 58(1):25–36, 2020.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[20] Carl H. White, George V. Lauder, and Hilary Bart-Smith. Tunabot Flex: a tuna-inspired robot with body flexibility improves high-performance swimming. *Bioinspiration & Biomimetics*, 16(2):026019, 2021.

[21] Yi Zhu, Fang-Bao Tian, John Young, James Liao, and Joseph Lai. A numerical study of fish adaption behaviors in complex environments with a deep reinforcement learning and immersed boundary–lattice Boltzmann method. *Scientific Reports*, 11:1691, 2021.

Authorized licensed use limited to: University of New South Wales. Downloaded on August 01,2025 at 01:50:21 UTC from IEEE Xplore. Restrictions apply.