CPSC 526: 2D Swimming Optimization

Ben Humberston*
University of British Columbia

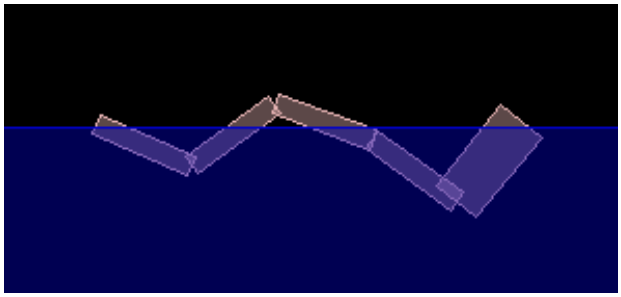


Figure 1: *Tadpole creature swimming using automatically learned control strategy*

Abstract

This paper describes the motivation, implementation process, and results of applying a covariance matrix adaptation (CMA) optimization to 2D rigid body characters that sit at the surface of a level fluid air interface. The optimization seeks to maximize swimming performance while optionally minimizing energy usage or angular motion of the character's body. Positive results are presented for two simple characters, with analysis of negative results for a more complex human character.

Keywords: fluid simulation, PD control, animation

1 Introduction

Aquatic locomotion control provides an interesting companion problem to that of ground-based locomotion. As noted by [Yang et al. 2004], unlike for walking and running, the character state (positions & velocities) during swimming is comparatively stable and smooth during the course of the stroke. As such, it provides a more well-behaved testing environment for evaluating control algorithms. On the other hand, the introduction of fluid dynamics to the rigid body simulation provides unique new challenges for building a controller.

In this work, we explore one method of designing the control of a virtual swimming character in two dimensions. A character composed of 2D rigid bodies is placed at the surface of a fluid environment that applies buoyancy and drag forces to the character. Two possible control strategies are implemented and compared. The first strategy learns the parameters for control functions, one per joint, which directly produce joint-driving torques. The second strategy, rather than directly learning desired torques, learns angle trajectory functions that are used to drive PD controllers at each joint. As described in Section 4, the second approach yielded more stable results for complex creatures. The control parameters for both strategies were chosen via the CMA technique as described in [Hansen and Kern 2004].

We discuss previous work for virtual swimming characters in Section 2. Section 3 gives details on how the simulation environment and virtual characters are defined. The two control strategies used in this work are described in Section 4, and the cost function that evaluates the quality of these controls is explained in Section 5. Section 6 briefly describes the CMA-ES algorithm and its method of optimizing the controller using a given cost function. The results of optimizing controllers for various creatures are given in Section 7, with a discussion of the project results in Section 8. Finally, a short overview of the code architecture of this project is given in Section 9

2 Related Work

In [Sims 1994], both the morphology and control system of procedurally-defined articulated creatures are evolved using a genetic algorithm. One of the optimization goals centers on the swimming capabilities of a creature in a simple drag-force fluid environment. [Tu and Terzopoulos 1994] outlines the creation of autonomous fish in a 3D virtual marine environment. [van de Panne and Eugene 1993] provides a 2D fish as a demonstration for target-following locomotion with a stochastically optimized controller.

[Yang et al. 2004] presents a detailed exploration of a human character swimming in a fluid environment. It addresses complications from the air-fluid boundary plane and shows results for various fluid viscosities. The controller for the swimmer is a multi-layer sys-

*e-mail: bhumbers@cs.ubc.ca

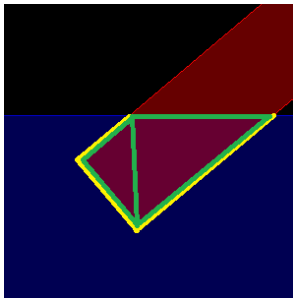


Figure 2: *Method for calculating buoyancy on a body. Once a list of submerged edges are determined (yellow lines), the total submerged area is found by summing the areas of each triangle defined by two adjacent submerged vertices and the point where the body intersects the fluid surface (green outlines). The buoyancy force is applied in the upward direction with magnitude proportional to the submerged area.*

tem which uses a top-level basic keyframed stroke and lower-level stroke modifications to achieve various tracking or velocity goals. [Tan et al. 2011] optimizes the swimming strategy of arbitrary articulated characters in a fluid environment using a realistic fluid simulation based on the Navier-Stokes equations as described in [Carlson et al. 2004]. Although this project uses a much simpler fluid model, we also employ the CMA optimization strategy from [Tan et al. 2011].

3 Simulation

The environment simulation was implemented using JBox2D, a Java port of the popular Box2D physics engine. The existing rigid body capabilities were supplemented with fluid buoyancy and drag, which creates forces on the bodies of characters. In this section we describe the details for character construction and simulating the fluid environment.

3.1 Character Specification

Each character is composed of a number of rigid bodies. It is assumed that the character is being viewed sidelong, so the different parts of the character at various depths will intersect on the 2D plane of projection. As such, the bodies that make up a character are free to intersect without penalty; physical motion constraints are imposed only by angle limits at the joints. Based on the given control strategy, a character may apply torques at each of the joints. The direct torque control strategy uses simple maximum torque limits, but trajectory-based controls set no limits on the torque produced by the associated PD controllers. Instead, low PD gains are used where possible to keep torques small and to allow reasonably sized simulation time steps.

3.2 Fluid Environment

The physical environment consists of a fluid pool half-plane that extends downward from the x-axis. Similar to [Yang et al. 2004], the internal fluid state is not explicitly modeled beyond a global average fluid velocity. The fluid applies both buoyancy and drag forces to submerged character bodies. An upward buoyancy force is applied at the centroid of the submerged portion of each body segment of a character, with the magnitude defined by

$$|F_b| = C_b \rho |g| \tag{1}$$

C_b is a user-defined buoyancy constant, ρ is the fluid density, and g is the magnitude of the gravitational force. The tessellation method used to find the submerged area of a body is illustrated in Figure 2.

A drag force is applied to the submerged portion of each body edge. The drag force is directed opposite the fluid velocity relative to the center point of the submerged edge segment. Its magnitude is set to

$$|F_d| = C_d l_n |v_n|^p \tag{2}$$

Here, C_d is a user-defined drag constant, l_n is the length of the submerged edge as projected onto a line perpendicular to the relative velocity of the fluid, v_n is the relative velocity of the fluid along the edge's normal, and p is an exponent set to either 1 or 2. Whether or not the $|v_n|$ term is raised to the first or second power depends on the fluid properties; this work uses the former value in all cases.

4 Control

Each character applies torques at its joints according to optimized control strategies. In contrast with the swimming human in [Yang et al. 2004] and the SIMBICON walker of [Yin et al. 2007], there are no manually-defined kinematic "keyframe" poses used as motion targets. Instead, we either directly learn torque controls or search for target trajectories that are followed using PD controllers at runtime. This keyframe-free strategy was chosen in order to explore whether desirable behavior could be produced without requiring explicit human hints to the correct swimming strategy in the form of target poses. We compare the two implemented control approaches in the following sections.

4.1 Direct Torque Control

In this approach, each joint applies a torque depending only on the current simulation runtime. The torque functions are composed of a number of basis functions whose parameters are optimized to produce different torque profiles. Gaussian and sinusoidal basis functions were implemented, with the type and number of basis functions fixed by joint. For joints that use n Gaussian bases, the jth basis function is defined by a weight coefficient A_j, a mean value μ_j, and a standard deviation σ_j:

$$\tau(t') = \sum_{j=1}^{n} A_j \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left(\frac{-(t'-\mu_j)^2}{2\sigma_j^2}\right) \tag{3}$$

$$t' = t \bmod T_{Gaussian} \tag{4}$$

To allow for cyclic torques, the simulation runtime t is transformed into t' by constraining its values to the time period $T_{Gaussian}$, where $T_{Gaussian}$ is a free parameter to be optimized, before being used as input to the torque control.

Alternatively, torque functions using sinusoidal basis terms have three parameters per term: amplitude A_j, period T_j, and phase offset ϕ_j:

$$\tau(t) = \sum_{j=1}^{n} A_j \sin\left(\frac{2\pi t}{T_j} + \phi_j\right) \tag{5}$$

The direct torque control strategy proved sufficient for a simple one-joint character, but did not scale well to more complex characters

like the tadpole or human. It is hypothesized that the fitness landscape for these characters when using direct torque control may be too rough to navigate without adding more intelligent constraints on torques. For example, coordinated swimming control may require the periods and amplitudes of torques for different joints to be related by a simple ratio to encourage synergies between joints.

4.2 Reference Trajectory Control

Due to the failure of direct torque control to scale to complex characters, a trajectory-based control strategy was also implemented. In this approach, the controller defines joint angle reference trajectories as a function of a phase input Φ. The simulation runtime t is generally used as the driving phase, though alternative phase sources may also be used (in the human character experiment, the angle of a shoulder is used as phase, as described in Section 7.3).

Polynomial and sinusoidal trajectory basis functions were implemented for this project. The latter is inspired by [Tan et al. 2011], which notes that a sine-based angle trajectory intuitively fits the cyclic nature of swimming strokes. For polynomial trajectories, given a manually chosen number of terms n, the term coefficients c_j are the free parameters to be optimized:

$$q_r(\Phi) = c_n\Phi^n + c_{n-1}\Phi^{(n-1)} + ... + c_2c_n\Phi^2 + c_1\Phi + c_0 \quad (6)$$

As with the direct torque controllers, sinusoidal reference trajectories are composed of n sine-function basis terms, each with the amplitude A_j, period T_j, and phase offset ϕ_j as free parameters:

$$q_r(\Phi) = \sum_{j=1}^{n} A_j \sin(\frac{2\pi\Phi}{T_j} + \phi_j) \quad (7)$$

During simulation, given a reference trajectory q_r for a joint and the current phase value Φ, a torque is applied to track the trajectory using PD control:

$$\tau(\Phi) = -k_p(q(\Phi) - q_r(\Phi)) - k_d\dot{q}(\Phi) \quad (8)$$

q and $\dot{q}$ are the current value of the joint angle and joint angular velocity. k_p and k_d are the proportional gain and velocity damping, respectively, and are chosen per character based on manual experimentation.

5 Cost Function

A simple cost function was created to guide the evolution of the control parameters in the optimization stage. Given a control strategy, a character runs in the simulated fluid environment for a user-specified period (generally 5 or 10 seconds); the value of the cost function is updated after each simulation time step. Aside from the primary goal of producing characters that move in the horizontal direction (along the surface of the water), the cost function contains terms that seek to minimize energy consumption and deviation of the characters root body from its original orientation:

$$E = w_1E_{speed} + w_2E_{disp} + w_3E_{energy} + w_4E_{orientation} \quad (9)$$

E_{speed} is the sum of the absolute deviations from a user-defined linear velocity for the character at each time step. E_{disp} is an alternative measure of locomotion fitness that can be used either as a replacement for or in conjunction with E_{speed}. It is set to the absolute difference between a target terminal position and the character's position at the conclusion of the simulation period. E_{speed} encourages the character to maintain as close a velocity to a given target velocity as possible, while E_{disp} encourages a final target displacement value regardless of the intermediate velocities.

E_{energy} is the sum of absolute torque magnitudes applied by the character across all joints during the simulation. $E_{orientation}$ is the sum of deviations of the character's orientation above some threshold value when compared to its original orientation at the start of simulation. The deviation threshold is used to avoid penalizing very small orientation deviations.

The combination of weight coefficients w_i are chosen based on each experiment, as described in Section 7.

6 Optimization

The control parameters for both the direct torque control and trajectory-based approaches were derived using the covariance matrix adaptation evolution strategy (CMA-ES). The Java implementation of this algorithm provided as a companion to [Hansen and Kern 2004] was employed using the recommended population size and other learning parameters. CMA-ES is a stochastic evolutionary scheme that samples the space of control parameters using a dynamically changing distribution. At each iteration, a group of control samples are evaluated for fitness using the cost function described in Section 5. The most successful samples are used to modify the covariance matrix and sample means of the distribution, influencing the probability of drawing similar samples in the next iteration. The best ever samples are remembered and returned once a maximum fitness or iteration limit is reached by the algorithm.

Generally, the number of CMA iterations required to find optimal values varied by experimental specifics, such as the character being controlled and the complexity of the cost function. Experiments using the simple paddle character and tadpole generally used fewer than 100 iterations before converging to satisfactory control strategy. The human character controller, however, failed to converge to acceptable fitness even after several hundred iterations. The time to run 300 iterations of CMA optimization for the human character's controller, the most complex control example in this project, ran in approximately 3 minutes on a machine with a 2 GHz dual-core processor with 4 GB memory.

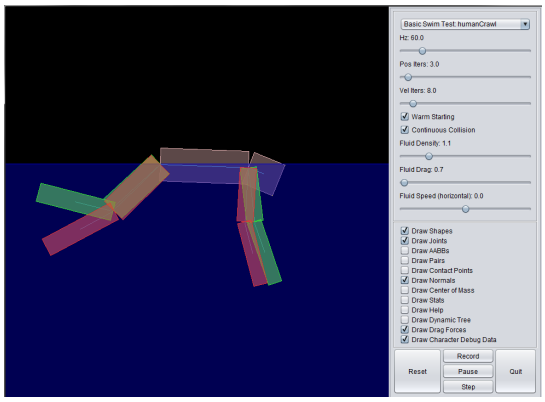
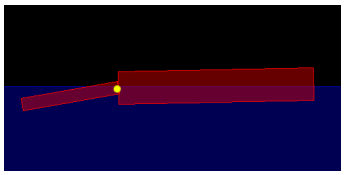


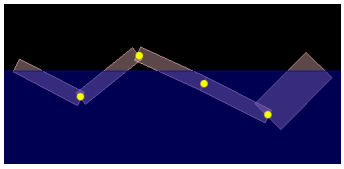
Figure 3: *GUI for testing optimized strokes in a fluid environment. The controls on the right side of the screen allow the user to select different characters, modify fluid properties, and change other simulation settings.*

Once the control parameters for a character are chosen using the CMA optimizer, they are saved to a comma-separated value file. The loaded parameters may then applied to the character in a provided graphical user interface application in order to view and interact with the results of the control policy in real-time. A screenshot of this application is shown in Figure 3.

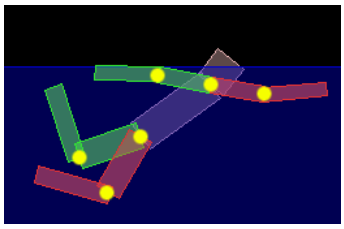
7 Evaluation



(a) *Paddle Character*



(b) *Tadpole*



(c) *Human*

Figure 4: *Characters used in this project. Yellow dots indicate joint positions*

Optimization experiments were run with three characters of varying complexity, as shown in Figure 4. The fluid density, buoyancy, and drag constants were set to a constant value of, respectively, $\rho = 1.1$, $C_b =$, and $C_d = 0.7$ across all experiments. The PD controller gain and damping values, k_p and k_d, were manually chosen for each experiment based on empirical testing. We describe the specifics for individual experiments below. Sequences demonstrating these experiments are included in the accompanying video (video link).

7.1 Simple Paddle

The paddle character consists of only a low-density deck and a higher density paddle body that rotates about a fixed joint at the back of the deck. In contrast to the tadpole and human characters, this character employs the direct torque controller approach, which is possible due to the simplicity of the character. It uses a combination of two Gaussian basis functions to determine the joint torque at a given time. Figure 5 illustrates the locomotion over time of three paddle creatures that were optimized solely based on different target speeds.

We also derived controllers that optimize energy efficiency (minimal torques) or for minimal deviation of the deck body's orientation by appropriately reweighting the cost function. In the torque minimization experiment, the faster characters maintain a reasonable speed while exerting significantly less torque than in the speed-focused experiment. However, the slowest character remains stationary rather than apply enough torque to move forward at a slow rate. In the experiment where the cost function encouraged minimal deck orientation deviations, the characters apply modest and slowly varying torques to avoid "tipping" the deck while still attempting to move at a target speed.

7.2 Tadpole

The tadpole character bears a resemblance to the recursive aquatic creatures in [Sims 1994]. A broad head segment is connected to a series of actuated tail bodies. Each joint uses a sine-based trajectory function, which results in cyclic motion of the tail.

For the tadpole experiments, only $E_{displacement}$ was considered, with the goal displacement for a 10 second simulation time set to 5 meters. Using the controller derived from this cost function, the tadpole whips each tail segment back and forth to produce forward thrust. We observed the performance of the tadpole controller after running 1, 5, and 50 CMA iterations. While the tadpole is able to drive itself forward at a low speed after only a single iteration, it achieves its top speed of about 0.6 m/s only after 50 iterations.

7.3 Human

The most complex control experiment uses a human-style character. The character is comprised of a head, a torso, and a pair each of arms and legs. While the head and appendages are dense enough to sink, the torso's density is roughly half the fluid density, which emulates the buoyancy of a torso when the lungs are filled with air. The character has 9 free joints: 1 for the neck where the torso meets the head, 2 for each arm (shoulder and elbow), and 2 more for each leg (hip and knee). Note that, similar to the joint from the paddle character, the shoulder is not angle-limited, so it may revolve continuously through a circle. This emulates the rotational capabilities available to a three-dimensional human. The elbow, hip, and knee joints used sine-based trajectories with reasonable joint limits.

In order to help stabilize the synchronization between different limbs, the human used a cyclic phase input to the reference trajectories rather than the linear time input used by the tadpole's trajectories. The phase value Φ was a mapping of the right shoulder joint's angle into the [0, 1] range. The trajectory of the right shoulder joint was manually chosen to increase linearly with time, but all other joint trajectories were optimized using Φ as input. Using this phase value rather than time drives the character to different poses based on the reference joint's current angle. If the phase-driving right shoulder lags behind its target trajectory, the other joints will maintain pace with it rather than moving ahead with time, a behavior which was hypothesized to increase stroke stability.

Unfortunately, no natural swimming motions were found for the human character using either direct torque motors or the optimized reference trajectories. The final human motions are erratic and do not consistently move in any one direction. Experiments showed that using the shoulder-locked phase helped prevent "loop-de-loop" motions of the character caused by out-of-synch joints, but the arms and legs still failed to work in tandem to move the character forward.

The accompanying video for this project includes a demonstration result of the final human control for the viewer's scrutiny and amusement.

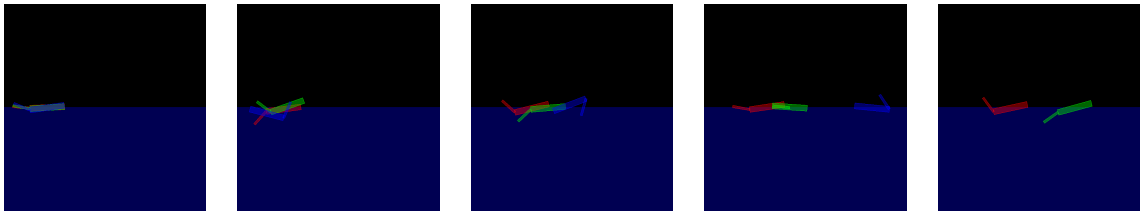


Figure 5: *Snapshots of three paddle characters optimized for different target speeds. The red, green, and blue characters are optimized to move at horizontal speeds 0.5, 2, and 5 m/s, respectively, though they are not able to attain the higher velocities. Note that the highest speed character, in blue, flips over onto its back before advancing past the other characters. See accompanying video for an animated version of this experiment.*

8 Discussion

The controller scheme with CMA optimization scheme provided positive results for simple characters, but did not find suitable strategies for complex morphologies. This failure to scale may be caused by poorly chosen optimization parameters, a discontinuous cost landscape with isolated minima, or other factors. While the optimization algorithm in this project may be able to *refine* an existing stroke solution, it is not able to derive a complete control strategy for high dimensional characters. This provides motivation to use manually defined key stroke poses as in [Yang et al. 2004], which helps narrow the control space at the cost of requiring manual pose construction.

Another difficulty encountered during implementation was finding a way to set manual hyper-parameters of the system, such as the PD gain and damping or the relative cost function term weights. Generally, these values needed to be tuned for each character specifically. Finding these values automatically would require a higher level optimization infrastructure that makes use of the outputs of the cost function and current CMA optimizer.

Finally, it should be noted that the fluid simulation used in this project was greatly simplified from the physical system it represents. Only drag and buoyancy fluid forces are modeled; fluid lift and friction forces were ignored, and internal fluid velocities are assumed to be zero at all points. The coupled rigid-fluid simulator in [Tan et al. 2011] demonstrates how using simplified fluid models may affect the learned control. Additionally, the two-dimensional planar representation required characters to use density values and joint limits that cannot be directly correlated to their three-dimensional equivalents. A future extension of this project may explore optimizing motion for three dimensional characters.

9 Code Reference

The following is a brief architectural layout of the source code for this project:

- *Entry Points*: The primary entry point for the GUI application to view swimming characters in realtime is found in *SwimGUIMain.java*. The control optimization algorithm is executed by running *SwimOptimizeMain.java* with one of the character versions as an input argument (eg: "tadpole", "humanCrawlRefTraj", etc.).
- *GUI*: The user interface is adapted from the JBox2D testbed application. In the GUI, a user may load pre-optimized characters. Fluid parameters such as buoyancy and drag may be modified in realtime. Relevant code is found in the *ubc.swim.gui* package.
- *Optimization*: The main CMA optimization loop is found in *ubc.swim.optimization.SwimmerOptimization*. The fitness function used for all the experiments in this work is defined in *ubc.swim.optimization.SwimFitnessFunctionA*.
- *Simulation*: The code that defines the characters, reference trajectories, and torque motors are found in the *ubc.swim.world* package and its subpackages. The buoyancy and fluid forces are applied by a force controller found in *ubc.swim.dynamics.controllers.FlulidController.java*.

References

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, SIGGRAPH '04, 377–384.

HANSEN, N., AND KERN, S. 2004. Evaluating the cma evolution strategy on multimodal test functions. *Parallel Problem Solving from Nature-PPSN VIII 7*, 282–291.

SIMS, K. 1994. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '94, 15–22.

TAN, J., GU, Y., TURK, G., AND LIU, C. K. 2011. Articulated swimming creatures. In *ACM SIGGRAPH 2011 papers*, ACM, New York, NY, USA, SIGGRAPH '11, 58:1–58:12.

TU, X., AND TERZOPOULOS, D. 1994. Artificial fishes: physics, locomotion, perception, behavior. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '94, 43–50.

VAN DE PANNE, M., AND EUGENE, F. 1993. Sensor-actuator networks. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '93, 335–342.

YANG, P.-F., LASZLO, J., AND SINGH, K. 2004. Layered dynamic control for interactive character swimming. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '04, 39–47.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: simple biped locomotion control. In *ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, SIGGRAPH '07.