Artificial jellyfish: evolutionary optimization of swimming

V. Lazunin
Hosei University
lazunin@gmail.com

V. Savchenko
Hosei University
vsavchen@hosei.ac.jp

ABSTRACT

Jellyfish, also known as "medusae", move by rhythmically contracting and expanding their bell-shaped bodies and are the earliest known animals to achieve locomotion through the muscle power. Development of a generalized dynamical model of medusan swimming is of interest to biologists as well as engineers. In this paper we present a new approach to modeling the swimming behavior of a jellyfish. Due to the axial symmetry of the creature we used a 2D cross-section for the calculation with the surface of the bell represented by two hemi-ellipsoidal curves. A simplified approach based on non-linear deformations of a geometric object is used to model the bell contraction-expansion cycle. We used a particle-gridless hybrid method for the analysis of incompressible flows, with averaging velocities field by the Shepard's method (partition of unity). To the best of our knowledge this is the first work where the optimal contraction and expansion parameters for the jellyfish movement were found by solving the optimization problem of maximizing the speed while minimizing the energy loss.

Keywords

Fluid dynamics, jellyfish, vortex, elasticity, optimization

1 INTRODUCTION

Jellyfish are the earliest known animals to use muscle power for swimming [DCC07]. They swim by contracting and expanding their mesogleal bells. The swimming muscles contract to expel a portion of water rearward out of the subumbrellar cavity, thus generating a thrust force to move the animal forward. The bell is refilled when it restores its shape after deformation it received during the thrust phase. The bell consists of a fiber-reinforced composite material called "mesoglea". The elastic characteristics of the mesogleal tissue were studied, for example, by Megill et al. [MGB05]

The contractile muscle fibers of the medusae are only one cell layer thick, so the forces that they can produce do not scale favorably with the increasing medusa size. For a medusa with the bell of diameter D, the mass of water that needs to be expelled from within the bell scales as D^3, while the muscle force only scales as D^1. Therefore the force required for jet propulsion increases with the animal size more rapidly than the available physiological force [DCC07]. Thus, the swimming performance may change dramatically with the increase of the medusan body size, and it is impossible to predict the optimal swimming parameters based on the geometric and kinematic similarity.

The physics of jellyfish swimming is not well understood. Existing animation techniques use combinations of sinusoidal curves to specify the deformations. However, it is important for animation to achieve a realistic movement depending on the size and shape of a bell. We assume that "realistic" also means "optimal", as the movements of the real jellyfish were "optimized" by the process of natural evolution, and we, therefore, would be able to find realistic movements for an artificial 3D model of jellyfish by means of artificial evolution. Other applications, such as computational biology, soft robotics and development of new propulsion techniques can benefit from development of a generalized model of jellyfish swimming.

In this paper we present a system for finding optimal swimming parameters for jellyfish models, based on our previous work where we studied vortex simulation for jellyfish [LS10]. The system consists of two main parts: simulated swimming and motion optimization. We introduce a simple technique based on radial basis functions (RBF) to model deformations of the jellyfish bell and a particle-gridless hybrid method for the analysis of incompressible flows. We modeled the interaction between the fluid particles and the surface of the bell in a form of elastic collision and reflection of the fluid particles off the boundary surface. The swimming efficiency was estimated for the bell and its particular movement specified by a set of control points. Genetic algorithms were used to find the optimal swimming pattern. To the best of our knowledge, this is the first work where the optimal swimming parameters for the jellyfish movement were found by solving the optimization problem. Throughout the paper we refer to two other paper concerning computational simulation of jellyfish ([LM09] and [RM09]), but neither of those employs any numerical optimization.

The remainder of the paper is divided into 7 sections. In section 2 we discuss related work. We describe our approach in sections 3 to 5 and outline the algorithm in section 6. In section 7 we outline the specifics of our prototype implementation and report of the experimental results, and in section 8 we conclude and describe directions of future work.

2 RELATED WORK

2.1 Studies of real life jellyfish

Experimental studies, including dye injection, filming and analyzing the resulting flow, indicate that smaller prolate medusae create strong jets during their bell contraction stage. Bigger oblate medusae, however, produce substantially less distinct jets and broad vortices at the bell margins. A hypothesis proposed by Colin and Costello [CC02] [DCC07] [DCC03] [DCCG05] is that oblate species are using their bell's margins as "paddles", thus utilizing a paddling, or rowing, mode of swimming. According to the model presented by Dabiri et al. [DCC07], big oblate medusae are not capable of swimming via jet propulsion. There is, however, a study of McHenry and Jed [MJ03] which suggests that the jetting model still provides more accurate approximation of swimming in oblate jellyfish.

The flow generated by oblate medusa's pulsatile jets consists mostly of radially symmetric rotating currents called vortex rings. To better understand the vortex formation and their effect on swimming performance, numerous experimental studies of real live jellyfish were performed [CC02] [DCC07] [MJ03] [DG03] [DCC03] [DCCG05]. Researches using mechanical jet generators demonstrate that there is a physical limit – called the "vortex formation number" – for the maximum size of the vortex rings. Once this number is reached, no bigger vortex formation is possible, and the extra water creates a trailing current behind the vortex. The energy cost for generating this current is higher than that of creating the vortex ring, so it is optimal to generate the largest possible vortex without any trailing current [DCC03]. Both thrust and efficiency increase in direct proportion with vortex ring volume [DCCG05]. Lipinski and Mohseni [LM09] used digitized motions of two real hydromedusae to computationally simulate the flows. Their results confirm the hypothesis proposed by Colin and Costello and demonstrate that distinct type of jellyfish ("jetting" and "paddling") produce substantially different kinds of vortices.

2.2 Fluid-solid interaction

Müller et al. proposed a particle-based method for interaction of fluids with deformable solids [MST$^+$04]. In their method they model the exchange of momentum between Lagrangian particle-based fluid model and solids represented by polygonal meshes with virtual boundary particles to model the solid-fluid interaction.

Lipinski and Mohseni [LM09] used digitized motions of two real hydromedusae to computationally simulate the flows. They used a new arbitrary Lagrangian-Eulerian method with mesh following the boundary between the fluid and the jellyfish body.

Yoon et al. presented a particle-gridless hybrid method for the analysis of incompressible flows [YKO99]. Their numerical scheme included Lagrangian and Eulerian phases. The moving-particle semi-implicit method (MPS) was used for the Lagrangian phase, and a convection scheme based on a flow directional local grid was developed for the Eulerian phase.

Chentanez et al. presented a method for simulating the two-way interaction between fluids and deformable solids [CGFO06]. The fluids were simulated using an incompressible Eulerian formulation where a linear pressure projection on the fluid velocities enforces mass conservation, whereas elastic solids were simulated using a semi-implicit integrator implemented as a linear operator applied to the forces acting on the nodes in Lagrangian formulation.

Hirato et al. proposed a method for generating animations of jellyfish with tentacles [HK03]. They used a simplified computational model based on the MPS method to simulate the fluid. Their work is mainly focused on visually plausible modeling of tentacles.

Rudolf and Mould created a system for physically-based animation of jellyfish [RM09]. Their approach may look very similar to ours, as they also exploited the radial symmetry, simulating only a 2D cross-section, and then creating a 3D bell for the visualization. The main difference between the approach proposed in [RM09] and the one discussed in this paper is that Rudolf and Mould did not employ any optimization, instead assigning a visually plausible set of parameters manually, by trial and error. They used a spring-mass system to represent the body of a jellyfish and a grid-based immersed boundary method for fluid-solid coupling. As they note in their work, there is still very little knowledge about physical properties of real jellyfish. Thus, we didn't feel necessary to employ something as complex as a spring-mass system, since the actual physical accuracy of the model would still be uncertain. Moreover, modeling a multi-layered structure of the jellyfish bell with only one layer of springs attached directly to the opposite sides of the bell does not look realistic. Some fugures from [RM09] demonstrate drastic change of both area and linear size of the umbrella cross-section during the contraction, something we failed to observe in real species, such as presented in experiments of Colin and Costello [DCCG05]. Instead of a spring-mass system, we use a simpler approach, with the umbrella of the jellyfish represented in 2D as two spline curves, deformed by RBFs. Instead of a grid-based method, for fluid simulation we used a particle-based method [YKO99] with elastic collision and reflection of the fluid particles off the boundary surface to prevent fluid leaking across the boundary. Finally, [RM09] employs a very primitive visualization technique, an issue we were trying to address with a GPU-based parallel ray tracer, capable of representing transparency, reflectivity and venous structure.

2.3 Optimization

The problem was studied by many researchers from the computer graphics and animation community, but we have no room for the comprehensive referencing, so we will mention only a few we found most relevant to out work.

Sims was one of the pioneers of artificial evolution. In his work [Sim91] he used genetic algorithms to create evolving images, textures, animations and plants, represented by procedural geometry, with human aesthetical selection instead of a fitness function. In [Sim94] he used similar approach to artificially evolve both morphology and behavior of articulated (e. g. composed of rigid parts and connecting joints) creatures, which were evolved and trained to perform specific tasks, like walking, jumping, following a light source, competing for a ball with other creatures etc.

Terzopoulos et al. [TTG94] modeled artificial fish as NURBS and spring-mass systems, using simulated annealing to find efficient moving patterns. Based on simulated sensory input, their fish could learn complex group behaviour, such as schooling, mating etc.

Tan et al. [TGTL11] used covariance matrix adaptation to find optimal swimming motion for fish, frog, turtle and even some fictional creatures, represented as articulated bodies; however, they stated that their simulation method is unsuitable for soft body creatures, such as jellyfish.

The works, discussed in this section, inspired our attempt to create a combined approach suitable for modeling and optimizing jellyfish. We emphasize that our work unites two themes of different research history: generation of time-dependent shapes and estimation of dynamical characteristics of the generated models.

3 BELL SIMULATION

To simulate the bell contraction-expansion cycle we used a simplified approach based on non-linear deformations of a geometric object. Because the model of jellyfish has radial symmetry, we used a 2D model (cross-section) with the surface of the bell represented by two hemi-ellipsoidal curves – the upper and the lower. For our model we used a piece-wise linear approximation with the initial number of nodes equal 40. A space mapping technique based on RBFs (see [SS01], and references therein) was used for non-linear approximation of shape deformations in numerous applications. Space mapping in R^n defines a relationship between each pair of points in the original model and the model after geometric modification. Let an n-dimensional region $\Omega \subset R^n$ of an arbitrary configuration be given, and let Ω contain a set of arbitrary control points $\{q_i = (q_1^i, q_2^i, ..., q_n^i) : i = 1,2,...,N\}$for the non-deformed object, and $\{d_i = (d_1^i, d_2^i, ..., d_n^i) : i = 1,2,...,N\}$for the deformed object. By assumption, the points q_i and d_i are distinct and given on or near the surface of each of two objects. The goal of the construction of the deformed object is to find a smooth mapping function that approximately describes the spatial transformation. The inverse mapping function can be given in the form

$$q_i = f(d_i) + d_i, \tag{1}$$

where the components of the vector$f(d_i)$ are volume splines interpolating displacements of initial points q_i (see Appendix for the details).

4 FLUID-SOLID COUPLING

Using a grid-based approach for jellyfish is possible, but poses a number of problems. Using a regular grid, as in [TGTL11] for an elastic body with varying thickness will result either in a huge computational overkill (if the grid is dense enough to accomodate the thin edges), or in a poor accuracy of the computation (if the grid is more sparse). Using an irregular grid, as in [LM09] requires solving a mesh warping/re-meshing problem. Solving Navier-Stokes equations with moving boundary is a hard problem. For simplicity, we chose a particle-based method. Particle-based methods became a de-facto standard for a class of problems where high precision is not required. For modeling we used almost the same scheme as proposed by Yoon, Koshizuka and Oka [YKO99]. They proposed a particle-gridless hybrid method for the analysis of incompressible flows, where tracing of virtual moving particles is used instead of solving nonlinear equations of velocity field. A particle interacts with other particles according to a weight function $w(r)$, where r is the distance between two particles. The weight function used by Koshizuka et al. is

$$w(r) = \begin{cases} -(2r/r_e)^2 + 2 & (0 \le r < 0.5r_e) \\ (2r/r_e - 2)^2 & (0.5r_e \le r < r_e) \\ 0 & (r_e \le r) \end{cases} \tag{2}$$

Density for a particle is calculated as the sum of weights of its interactions with the other particles (all interaction happens only within the radius r_e):

$$\langle n \rangle_i = \sum_i w(|r_j - r_i|). \tag{3}$$

Note, that, unlike in the MPS method, the particle number density here is not required to be constant. A gradient vector between two particles i and j possessing scalar quantities ϕ_i and ϕ_j at coordinates r_i and r_j is equal to $(\phi_j - \phi_i)(r_j - r_i)/|r_j - r_i|^2$. The gradient vector at the particle i is given as the weighted average of these gradient vectors:

$$\langle \nabla \phi \rangle_i = \frac{d}{n^0} \sum_{j \neq i} \left[\frac{\phi_j - \phi_i}{|r_j - r_i|^2} (r_j - r_i) w(|r_j - r_i|) \right], \quad (4)$$

where d is the number of space dimensions and n^0 is the particle number density.

Diffusion is modeled by distribution of a quantity from a particle to its neighbors using the weight function:

$$\langle \nabla^2 \phi \rangle_i = \frac{2d}{\lambda n^0} \sum_{j \neq i} [(\phi_j - \phi_i) w(r_j - r_i)], \quad (5)$$

where λ for a two-dimensional case with Equation (2) as the weight function is equal to $\frac{31}{140} r_e^2$. This model is conservative, because the quantity lost by the particle i is obtained by the particle j.

The continuity equation for incompressible fluid can be written as follows:

$$\frac{D\rho}{Dt} = -\rho(\nabla \cdot u) = 0. \quad (6)$$

The velocity divergence at the particle i is given by:

$$\langle \nabla \cdot u \rangle = \frac{d}{n^0} \sum_{j \neq i} \frac{(u_j - u_i) \cdot (r_j - r_i)}{|r_j - r_i|^2} w(|r_j - r_i|). \quad (7)$$

Then the pressure is calculated as:

$$\frac{u_i^{**} - u_i^*}{\Delta t} = -\frac{1}{\rho} \langle \nabla P^{n+1} \rangle_i, \quad (8)$$

$$\langle \nabla^2 P^{n+1} \rangle_i = \frac{\rho}{\Delta t} \langle \nabla \cdot u^* \rangle_i, \quad (9)$$

where u^* is the temporal velocity obtained from the explicit calculation and u_i^{**} is the new-time velocity. The left side of (9) is calculated using the Laplacian model (5). The right side is the velocity divergence, calculated by (7). We use variable r_e to avoid cases where some particles near the boundary will have very few neighbours to interact with. It gives a system of linear equations represented by an unsymmetric matrix, which is solved by an unsymmetric-pattern multifrontal method [Dav04]. Solving (9) may seem computationally expensive, but with jellyfish, the most important fluid-solid interaction often happens near the very thin edges of the bell, so calculating accurate pressure field is necessary.

Instead of using a higher-order gridless convection scheme as it was proposed by Yoon et al. [YKO99] to approximate flow directions, we applied averaging of the velocities field by a simple scheme, based on Shepard's method (partition of unity) [She68].

Boundary conditions are perhaps the most important factor influencing the accuracy of the flow computation. The manner in which the boundary conditions are imposed influences the convergent properties of the solution. Usually in particle-based methods boundary particles are used to approximate the no-penetration condition [MST+04] [PTB+03]. Repulsion and adhesion forces between the particles are used to simulate the no-penetration, no-slip and actio = reactio conditions on the boundary of the solid.

In our work contour points represent the geometry of the model and also define fluid boundaries. That is, the solution points are defined by the fluid particles and the particles located on the boundary of the bell. For each boundary particle we can calculate the boundary normal vector, pointing outwards, into the flow domain. For the no-slip condition, only the normal speed component of any boundary particle is used, while the tangential speed component is discarded. The no-penetration condition is modeled in a form of elastic collision and reflection of the fluid particles off the boundary surface. The motion of the bell was computed using only translational parts in y direction. One component of the force F on a rigid body is a derivative of linear momentum mv of the gravitational center. It is assumed that jellyfish body density is equal to the density of the water. Thus m is a volume occupied by the jellyfish.

The force F also invokes fluid and rigid body interaction. Points on the curve used to represent the bell can be considered as rigid particles. When the bell is deformed, distances between boundary particles may change, so we put a new set of boundary particles after each deformation, by evenly subdividing the curves. The strategy of using rigid particles we followed was first proposed in [CMT04]. The forces on rigid particles are computed by assuming the rigid body as a fluid. Therefore, for a particle i with the pressure p_i, mass density ρ_i and speed v_i, the force from the fluid acting on the node particle $f_i^{fluid} = f_i^{press} + f_i^{vis}$ is calculated by using the physical values of the neighbor particles as follows [DC96]:

$$f_i^{press} = -\sum_{i \neq j} (p_i + p_j)/2\bar{\rho}_j \nabla_i w_h^{ij} \quad (10)$$

$$f_i^{vis} = -\sum_{i \neq j} \Pi_{ij} \nabla_i w_h^{ij} \quad (11)$$

where

$$\Pi_{ij} = \begin{cases} -(c\mu_{ij} + 2\mu_{ij}^2)/\bar{\rho}_j & \mu_{ij} < 0 \\ 0 & \mu_{ij} \geq 0 \end{cases} \quad (12)$$

$\mu_{ij} = e_r v_{ij} r_{ij}/(r_{ij}^2 + 0.01 e_r^2)$, $\bar{\rho}_j = 0.5(\rho_i + \rho_j)$, $r_{ij} = r_i - r_j$, $v_{ij} = v_i - v_j$

5 OPTIMIZATION

In techniques based on the error functional minimization it may become necessary to solve highly non-linear problems. Minimization by standard techniques requires high computational effort. Minimization of a simplified functional, for example a quadratic one, is reduced to solving a simple system of linear equations. However, it leads to iterative minimization that depends on a sufficiently good initial guess. It seems to us that an attractive way of attacking this problem is to use optimization techniques based on genetic algorithms, proposed by Mahfoud and Goldberg in [MG92]. In this work we used an algorithm with simulated annealing type selection.

The application of the genetic algorithm starts with initially selecting a set of M variable control points $\{d_i = (d_{1i}, d_{2i}, d_{3i}, ...) : i = 1, 2, ..., M\}$ for the definition of the space transformation generating the deformed object. Actually, the user defines points q_i on the initial image of the bell in its rest state with corresponding points d_i on the model of fully contracted bell (Fig. 1). The collection of coordinates d_i and the contraction time t_{cont} define a creature. The algorithm begins by randomly distorting the initial creature and generates s creatures, which form the initial population. Now, the genetic algorithm with sequential simulated annealing is applied to this initial population to minimize the fitness function.

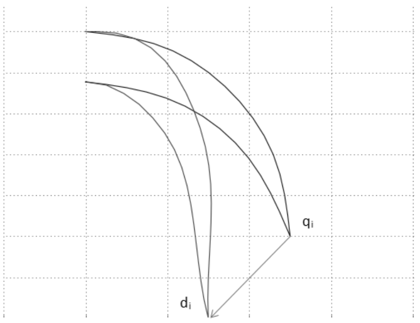

Figure 1: An example of a creature: the right half of the bell cross-section in two states (initial and deformed) and the deformation vector

The spline $f(P)$, determined by the set of N variable control points d_i which constitute a creature, used for global space mapping, provides a minimization of quantity $h^t A^{-1} h$, that is called "bending energy". 8 points belonging to the border of the bounding box and two additional points in the center of the bell ($x = 0$) on the upper and lower curve are used as anchor points. k destination points define general deformation of the jellyfish bell. A^{-1} is the bending energy matrix, which is the inverse $N \times N$ upper left submatrix of T, and h_i are so-called heights and $N = 10 + k$. Space transformation h_i is the difference between the coordinates of the initial and destination point placements as shown in Fig. 1. The bending energy of a general transformation is the sum $x^t A^{-1} x + y^t A^{-1} y$ of the bending energy of its horizontal x-components, modeled as a "vertical" plate, and the bending energy of its vertical y-component, modeled similarly as a "vertical" plate.

In our simulation we used the "economy" principle: the jellyfish is striving to reach maximum speed with minimum deformation of the bell. Thus, one of the fitness function component is the bending energy E^b. In the numerical analysis we also measured two quantities characterizing jellyfish locomotion, i. e. distance D passed by a body which is defined by swimming speed v and energy loss E^l. Energy loss is assumed to be equal to surrounding water energy. Following the "economy" principle, we define the fitness function as follows:

$$Fitness = w_v \cdot D - w_b \cdot E^b - w_e \cdot E^l \qquad (13)$$

where w_b is the weight for RBF energy, w_e is the weight for kinetic energy of a particle, w_d is the weight for the object velocity. These parameters are set by the user to choose a mode of movement.

6 ALGORITHM

Initially, the 2D contour of the bell cross-section is specified as an array of points. Deformations are assigned to the bell margin points. Particles are placed at a regular interval (on a regular grid) inside the bounding box, except the inner area of the bell. Then, the following steps are performed iteratively for each step of the contraction/expansion cycle:

1. The averaged density is calculated for every particle. A ball is generated for every particle, and the density is defined as the volume of the ball divided by the number of particles inside the ball. The ball diameter is not constant, and is adjusted so that all balls contain roughly similar number of particles.

2. The bell margin points are moved by a step along the deformation vectors. For the rest of the points their displacement vectors are calculated using RBFs.

3. A cardinal spline is fit through the displaced boundary points. Because some segments may become too long, we discard the boundary points and insert them again by subdividing the spline curve evenly, so that all the distances between neighboring points are mostly equal.

4. A Poisson equation in a matrix form (unsymmetric, about 10000 linear equations) is solved, giving new values of pressure for each particle.

5. Gradient vectors are calculated applying equation (4). For every particle, the speed vector is calculated, and the particle is then moved along the vector by the time step Δt.

6. New pressure values for the displaced particles are interpolated back to the nodes of the regular grid.

Number of sub-steps	Time (sec)	Path (m)
6	7	0.02
10	13.14	0.032
14	18	0.031
18	23.99	0.033
20	26.6	0.038
40	25.39	0.039

Table 1: Performance results

7. Distance passed and energy lost at this step are calculated for the creature.

At the end of a swimming cycle, we have a fitness for the creature according to (13). A population of 10 such creatures is evolved until convergence within 10%. The best creature is then selected as "optimal".

For animation, we created a 3D model out of the 2D contour, and then visualized by ray tracing. At first, the mesh was created by rotational extrusion and tessellation of the original 2D contour. Finally, we used Blender 3D modeling suite to create a roughly similar 3D model with some embellishments, such as inner "veins", inward-oriented "velum" and several tentacles. Rotation of the original 2D contour was still used to put anchor points at some interval around the bell. The anchor points were used as a "skeleton" for RBF-based deformations of the entire bell model in 3D, including the veins, at each animation step. Some random perturbations were added to the anchor points to make our jellyfish look less artificial (Fig. 2). The tentacles were modeled as soft cloth with one side attached to the bell and deformed separately. The cloth structure was represented by a triangular mesh, with nodes affected by the water flow.

7 RESULTS AND DISCUSSION

We implemented this method and used it to find the optimal swimming parameters for a simple oblate jellyfish similar to Aurelia aurita [DCCG05]. The program was written in C++ and CUDA C. UMFPACK library [Dav04] was used for solving large sparse systems of linear equations. The algorithm terminates when a satisfactory fitness level has been reached for the population. In practice it happens when the number of generations is approximately 30. As Table 1 shows, the path differs at the 3rd decimal digit. In our implementation we typically used 14 substeps with the calculation time of about 10 minutes on a single core of an Intel Core 2 Quad computer. The size of the simulation area is 20x20 cm, with the resoultion of 100x100 particles. The bell diameter is 10 cm.

In the particular case (see Fig. 2) weights (lazy mode) $w_b = 0.3$, $w_e = 0.3$ and $w_d = 0.4$ were used. The vortices produced by the simulation (Fig. 3) were similar to those observed for real jellyfish, showing that application of the no-slip condition looks reasonable.

A GPU-based parallel ray tracing system was developed for the visualization (Fig. 2). We used recursive ray tracing to visualize the bell as a transparent object with refraction and reflection. Each primary ray hitting the bell was spawning several secondary rays, so the animation performance varied depending on the number of primary rays hitting the bell, and, thus, on the distance from the camera to the jellyfish and on the viewing angle. Our ray tracer produced 20-30 frames per second on a GeForce GT 540M GPU. See the supplementary video for an example of animation. We must add, that experience in areas such as 3D art and texture painting would add significantly to the observed realism of the animation, but that is beyond the scope of this work.

To validate our results, we compared them with experimental data received by Colin and Costello [CC02] [DCCG05]. The distance passed by our model during one full contraction-expansion cycle (which is roughly equal to the bell radius) and the resulting water flows seem to be in agreement with their data.

8 LIMITATIONS AND FUTURE WORK

We employed our simulation software to find the optimal movement for a very simple 2D model, swimming straight ahead. A more thorough validation of our technique, using a variety of sizes and shapes, as well as robustness and sensitivity studies are required and are subjects of future work. A few control points were used to specify bell contraction, and the movements of the bell margins were set by only two axially-symmetric vectors. Real jellyfish have no brain or eyes (although some of them have photosensitive spots on their bells) and do not deliberately choose any complex swimming trajectory, so we think our simplification has no big impact on the results veracity for jellyfish. However, following complex paths would be crucial for jellyfish-like robots. To simulate such behavior, it may be necessary to perform the simulation in 3D and with larger number of (possibly asymmetric) control points.

We did not, either, take into account jellyfish feeding behavior and tentacles. Real jellyfish have an oral opening inside the bell. Some of them also have numerous tentacles spread in the water. The tentacles add drag force and decrease swimming performance, but are used to catch prey. Jellyfish create water flows to carry their prey through the tentacles or into the oral cavity itself. Modeling such behavior is important for computational biology. Additional parameters for it can be incorporated into the fitness function.

Another possible future work would be optimization of the shape itself, e. g. finding both optimal movement and optimal shape, satisfying constraints imposed by a 3D designer.

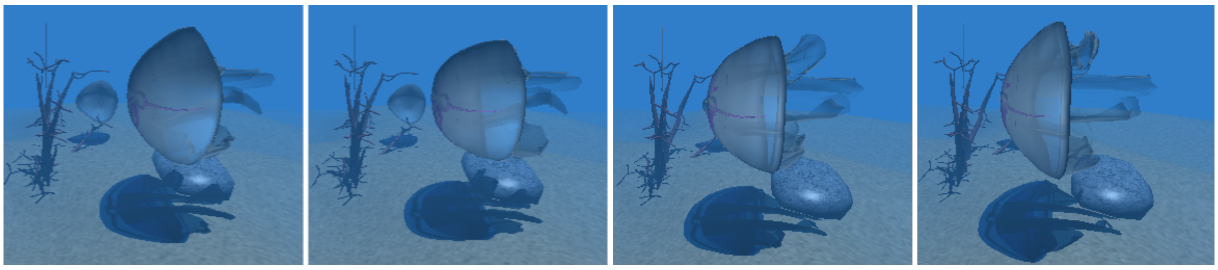

Figure 2: Animation sequence of two contraction steps (left) and two expansion steps (right) for a jellyfish with floating tentacles and small additional deformations applied to the bell.

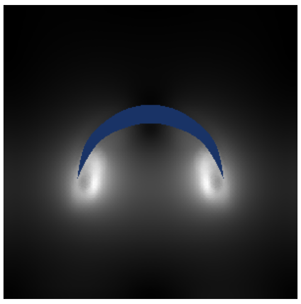

Figure 3: Vortex formation

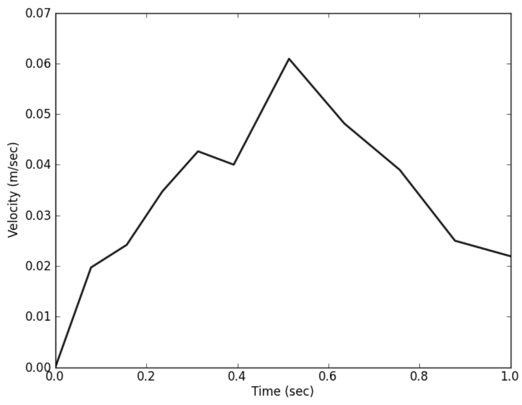

Figure 4: Velocity change for a jellyfish during the initial swimming cycle (10.00 cm diameter, 0.39 sec contraction time, 0.61 sec expansion time)

9 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful comments.

10 APPENDIX

We consider a mapping function as a thin-plate interpolation. For an arbitrary area Ω, the thin-plate interpolation is a variational solution that defines a linear operator T when the following minimum condition is used:

$$\int_{\Omega} \sum_{|\alpha|=m} m!/\alpha!(D^{\alpha}f)^2 d\Omega \rightarrow min, \tag{14}$$

where m is a parameter of the variational function and α is a multi-index. It is equivalent to using the RBFs $\phi(r) = rlog(r)$ or r^3 for $m = 2$ and 3 respectively, where r is the Euclidean distance between two points.

The volume spline $f(P)$ having values h_i at N points P_i is the function

$$f(P) = \sum_{j=1}^{N} \lambda_j \phi(|P - P_j|) + p(P), \tag{15}$$

where $p = v_0 + v_1 x + v_2 y + v_3 z$ is a degree-one polynomial. To solve for the weights λ_j we have to satisfy the constraints h_i by substituting the right part of Equation (15), which gives

$$h_i = \sum_{j=1}^{N} \lambda_j \phi(|P_i - P_j|) + p(P_i). \tag{16}$$

λ and v are the coefficients that satisfy a linear system $Tx = b$, where

$$\begin{aligned} T &= \begin{bmatrix} A & B^T \\ B & D \end{bmatrix}, \\ x &= [\lambda_1, \lambda_2, ..., \lambda_N, v_0, ..., v3]^T, \\ b &= [h_1, h_2, ..., h_N, 0, 0, ..., 0]^T \end{aligned} \tag{17}$$

For 2D and 3D cases we call $f(P)$ a volume spline.

REFERENCES

[CC02] S. P. Colin and J. H. Costello. Morphology, swimming performance and propulsive mode of six co-occuring hydromedusae. *The Journal of Experimental Biology*, 206:427–437, 2002.

[CGFO06] N. Chentanez, T. G. Goktekin, B. E. Feldman, and J. F. O'Brien. Simultaneous coupling of fluids and deformable bodies. *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, pages 83–89, 2006.

[CMT04] M. Carlson, P. J. Mucha, and G. Turk. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics*, volume 23, pages 377–384, 2004.

[Dav04] T. A. Davis. Umfpack, an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.

[DC96] M. Desburn and M. P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. *Computer animation and simulation*, pages 61–67, 1996.

[DCC03] J. O. Dabiri, S. P. Colin, and J. H. Costello. Fast-swimming hydromedusae exploit velar kinematics to form an optimal vortex wake. *The Journal of Experimental Biology*, 206:3675–3680, 2003.

[DCC07] J. O. Dabiri, S. P. Colin, and J. H. Costello. Morphological diversity of medusan lineages constrained by animal-fluid interactions. *The Journal of Experimental Biology*, 210:1868–1873, 2007.

[DCCG05] J. O. Dabiri, S. P. Colin, J. H. Costello, and M. Gharib. Flow patterns generated by oblate medusan jellyfish: field measurements and laboratory analyses. *The Journal of Experimental Biology*, 208:1257–1265, 2005.

[DG03] J. O. Dabiri and M. Gharib. Sensitivity analysis of kinematic approximations in dynamic medusan swimming models. *The Journal of Experimental Biology*, 206:3675–3680, 2003.

[HK03] J. Hirato and Y. Kawaguchi. Calculation model of jellyfish for simulating the propulsive motion and the pulsation of the tentacles. *18th International Conference on Artificial Reality and Telexistence*, 2003.

[LM09] D. Lipinski and K. Mohseni. Flow structures and fluid transport for the hydromedusae Sarsia tubulosa and Aequorea victoria. *The Journal of Experimental Biology*, 212:2436–2447, 2009.

[LS10] V. Lazunin and V. Savchenko. Vortices formation for medusa-like objects. *Proceedings of Fifth European Conference on Fluid Dynamics (ECCOMAS CFD 2010)*, June 2010.

[MG92] S. W. Mahfoud and D. E. Goldberg. A genetic algorithm for parallel simulated annealing. *Parallel problem solving from nature*, 2:301–310, 1992.

[MGB05] W. M. Megill, J. M. Gosline, and R. W. Blake. The modulus of elasticity of fibrillin-containing elastic fibres in the mesoglea of the hydromedusa polyorchis penicillatus. *The Journal of Experimental Biology*, 208:3819–3834, 2005.

[MJ03] M. J. McHenry and J. Jed. The ontogenetic scaling of hydrodynamics and swimming performance in jellyfish (aurelia aurita). *The Journal of Experimental Biology*, 206:4125–4137, 2003.

[MST$^+$04] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross. Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds*, 15:159–171, 2004.

[PTB$^+$03] S. Premože, T. Tasdizen, J. Bigler, A. Lefohn, and R.T. Whitaker. Particle-based simulation of fluids. *Computer Graphics Forum*, 22(3):401–410, 2003.

[RM09] D. Rudolf and D. Mould. Interactive jellyfish animation using simulation. *International Conference on Computer Graphics Theory and Applications (GRAPP)*, pages 241–248, 2009.

[She68] D. Shepard. A two-dimensional interpolation function for irregularly spaced data. *Proceedings of the 23th Nat. Conf. of the ACM*, pages 517–523, 1968.

[Sim91] K. Sims. Artificial evolution for computer graphics. *Computer graphics*, pages 319–328. ACM SIGGRAPH, July 1991.

[Sim94] K. Sims. Evolving virtual creatures. *Computer graphics*, pages 15–22. ACM SIGGRAPH, July 1994.

[SS01] V. Savchenko and L. Schmitt. Reconstructing occlusal surfaces of teeth using genetic algorithm with simulated annealing type selection. *6th ACM Symposium on Solid Modeling and Applications*, pages 39–46, June 2001.

[TGTL11] J. Tan, Y. Gu, G. Turk, and C. K. Liu. Articulated swimming creatures. *Computer graphics*, volume 30. ACM SIGGRAPH, July 2011.

[TTG94] D. Terzopoulos, X. Tu, and R. Grzeszczuk. Artificial fishes: autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, 1(4):327–351, 1994.

[YKO99] H. Y. Yoon, S. Koshizuka, and Y. Oka. A particle-gridless hybrid method for incompressible flows. *International Journal for Numerical Methods in Fluids*, 30:407–424, 1999.