Automated Learning of Muscle-Actuated Locomotion Through Control Abstraction

Radek Grzeszczuk and Demetri Terzopoulos
Department of Computer Science, University of Toronto [1]

Keywords: *learning, locomotion, control, artificial life, physics-based modeling*

Abstract: *We present a learning technique that automatically synthesizes realistic locomotion for the animation of physics-based models of animals. The method is especially suitable for animals with highly flexible, many-degree-of-freedom bodies and a considerable number of internal muscle actuators, such as snakes and fish. The multilevel learning process first performs repeated locomotion trials in search of actuator control functions that produce efficient locomotion, presuming virtually nothing about the form of these functions. Applying a short-time Fourier analysis, the learning process then abstracts control functions that produce effective locomotion into a compact representation which makes explicit the natural quasi-periodicities and coordination of the muscle actions. The artificial animals can finally put into practice the compact, efficient controllers that they have learned. Their locomotion learning abilities enable them to accomplish higher-level tasks specified by the animator while guided by sensory perception of their virtual world; e.g., locomotion to a visible target. We demonstrate physics-based animation of learned locomotion in dynamic models of land snakes, fishes, and even marine mammals that have trained themselves to perform "SeaWorld" stunts.*

1 Introduction

The animation of animals in motion is an alluring but difficult problem. With the advent of visually realistic models of humans and lower animals, even small imperfections in the locomotion patterns can be objectionable. The most promising approach to achieving a satisfactory level of authenticity is to develop physically realistic artificial animals that employ internal actuators, or muscles, to closely approximate the movements of natural animal bodies. As these animal models become increasingly complex, however, animators can no longer be expected to control them manually. Sophisticated models must eventually assume the responsibility for their own sensorimotor control. Like real animals, they should be capable of learning to control themselves.

This paper addresses a natural question: Is it possible for a physics-based, muscle-actuated model of an animal to learn from first principles how to control its muscles in order to locomote in a natural fashion? Unlike prior work on motion synthesis, we target state-of-the-art animate models of at least the level of realism and complexity of the snakes and worms of Miller [8] or the fish of Tu and Terzopoulos [17]. In both of these cases, the muscle controllers that produce locomotion were carefully hand crafted using knowledge gleaned from the biomechanics literature [7] and long hours of experimentation. Our goal in this paper is to devise algorithms that can provide such animal models the ability to learn how to locomote automatically, in a way that is inspired by the remarkable ability of real animals to acquire locomotion skills through action and perception.

At the foundation of our approach lies the notion that natural locomotion patterns are energetically efficient. This allows us to formalize the problem of learning realistic locomotion as one of optimizing a class of objective functionals, for which there are various solution techniques. We formulate a bottom-up, multilevel strategy for learning muscle controllers. At the early stages of the learning process, the animate model has no *a priori* knowledge about how to locomote. It is as if the animal had a fully functional body, but no motor control center in its "brain". Through practice—repeated locomotion trials with different muscle actions—the animal learns how to locomote with increasing effectiveness, by remembering actions that improve its motor skills as measured by the objective functional. Repeated improvements eventually produce life-like locomotion.

When basic locomotive skill is achieved, the animate models abstract the low-level muscle control functions that they have learned and train themselves to perform some specific higher-level motor tasks. The learning algorithm abstracts detailed muscle control functions into a highly compact representation. The representation now emphasizes the natural quasi-periodicities of effective muscle actions and makes explicit the coordination among multiple muscles that has led to effective locomotion. Finally, the artificial animals can put into practice the compact, efficient controllers that they have learned in order to accomplish the sorts of tasks that animators would have them do.

We are particularly interested in realistic motion synthesis for three dimensional models of animals that are highly deformable and can move continuously within their virtual worlds. Plates 1–5 show frames from animations of animal models that we have created, which have learned to locomote and perform interesting motor tasks automatically. We use spring-mass systems to construct our models, following the work of [8, 17]. This results in biomechanical models with numerous degrees of freedom and many parameters to control. The reader should peruse [8, 17] to become familiar with the details of the models.

An example will serve to illustrate the challenges of controlling highly deformable body models: Fig. 1 illustrates a biomechanical model of a Sonoral coral snake [4] that we use in one of our animations (Plate 1). The body consists of 10 segments. Each segment has two pairs of longitudinal muscle springs that are under the active control of the snake's brain. All other springs are passive dynamic elements that maintain the structural integrity of the body. The snake can actuate its body by varying the rest lengths of the 40 muscle springs over time. To simplify matters slightly, paired muscles on either side in each segment are actuated synchronously, and this yields a total of 20 actuators. Clearly, it is counterproductive to provide the animator direct control over so many actuators. Instead, we would like the snake to train itself to control its body. We will

[1] 10 King's College Road, Toronto, Ontario, Canada, M5S 1A4
E-mail: {radek|dt}@cs.toronto.edu

Published in the Proceedings of SIGGRAPH 95 (Los Angeles, CA, August 6–11, 1995). In *Computer Graphics* Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 63–70.

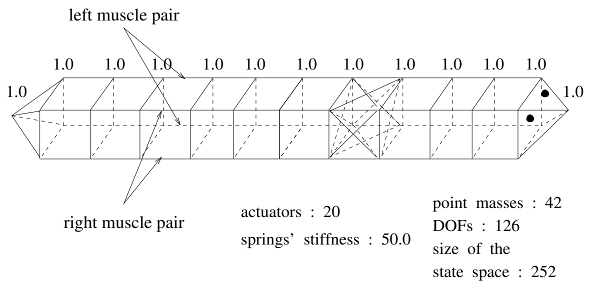


Figure 1: *The snake biomechanical model consists of nodal masses (points) and springs (lines). It has twenty independent actuators (muscle springs): ten on the left side of the body and ten on the right side. Each actuator comprises a pair of synchronous muscles. The numbers along the body indicate nodal masses in cross sectional planes. The cross-springs, shown in only one segment, maintain the structural integrity of the body.*

develop algorithms that will enable its brain to exercise its body until it discovers the actuator coordination needed to achieve efficient serpentine locomotion. The snake will monitor the progress of the learning cycle using an objective functional that incorporates sensory feedback about its actions.

An advantage of our approach from the point of view of the animator is its generality. In principle, it is applicable to all animate models motivated by internal muscles, whether highly deformable, or articulate. In this paper, we demonstrate its power using 4 different, highly deformable animal body models in varied media—terra firma, water, and air (see Appendix A). Another advantage is that the approach allows us to equip our models with sensors that enable them to perceive their environment. Sensory perception is modeled through the objective functional to be optimized. The sensory contribution to the objective functional represents the animal's perception of the degree to which its goal has been achieved. Making the artificial animal perceptually aware allows it to handle tasks that depend on dynamic events in the environment and gives the animator a potent tool with which to control the model.

1.1 Related Work

The issue of control is central to physics-based animation research. Optimal control methods formulate the control problem in terms of an objective functional which must be minimized over a time interval, subject to the differential equations of motion of the physical model [1]. The "spacetime constraints" method [19] has attracted a certain following (e.g., [3]), but it is problematic because, in principle, it treats physics as a penalty constraint (that can be "stretched like a spring") and, in practice, the need to symbolically differentiate the equations of motion renders it impractical for all but the simplest physical models.

We pursue a different approach toward locomotion control that is suitable for complex physical models. The approach is inspired by the "direct dynamics" technique which was described in the control literature by Goh and Teo [5] and earlier references cited therein. Direct dynamics prescribes a generate-and-test strategy that optimizes a control objective functional through repeated forward dynamic simulation and motion evaluation.

The direct dynamics technique was developed further to control articulated musculoskeletal models in [10] and it has seen application in the mainstream graphics literature to the control of planar articulated figures [18, 9]. Pandy *et al.* [10] search the model actuator space for optimal controllers, but they do not perform global optimization. Van de Panne and Fiume [18] use simulated annealing for global optimization. Their models are equipped with simple sensors that probe the environment and use the sensory information to influence control decisions. Ngo and Marks' [9] stimulus-response control algorithm presents a similar approach. They apply the genetic algorithm to find optimal controllers. The genetic algorithm is also used in the recent work of Sims [15]. Risdale [14] reports an early effort at controller synthesis for articulated figures from training examples using neural networks.

A characteristic of prior methods that tends to limit them to relatively simple planar models with few actuators is that they attempt to tackle the control problem at only a single level of abstraction. Typically, they deal with the control problem at an abstract level, say, in terms of a small number of controller network weights [18, 15] or whole body motions [9]. We advocate a multilevel controller learning technique that can handle complex models even though it seeks, based on first principles, optimal muscle actuation functions in a very concrete representation that makes the weakest possible assumptions. Thus the learning process is bootstrapped essentially from scratch. Earlier versions of our work were presented in [6, 16].

1.2 Overview

We describe our multilevel learning technique in the following two sections. Section 2 presents the strategy for learning low level controllers. Low level control learning is time consuming because of the high dimensionality of the search space. It is therefore prudent to reuse controllers. To this end, Section 3 presents the strategy for abstracting high level controllers. The abstraction step dramatically reduces dimensionality, stores the reduced description in the animal's memory, and permits the control problems to be defined in terms of higher level motor goals. This approach leads naturally to reusable solutions. We search for good low level control solutions for a set of simple tasks and use them as building blocks to achieve higher level goals. Section 4 presents a thorough experimental evaluation of our learning approach and more results.

2 Learning Low Level Control

Our low-level learning technique repeatedly generates a controller, applies it to drive a short-time forward simulation of the dynamic body model, and measures its effectiveness at producing locomotion using an objective functional. Typically, this low-level motor learning cycle is lengthy (as it can be in real animals, such as humans). However, it is simple and ultimately quite effective.

2.1 Biomechanical Models, Muscles, Actuators, Controllers

The *biomechanical models* that we employ are constructed of nodal masses and springs, as is detailed in Appendix A. Their dynamics is specified by the Lagrangian equations of motion

$$m_i \ddot{\mathbf{x}}_i + \gamma_i \dot{\mathbf{x}}_i + \sum_{j \in N_i} \mathbf{f}_{ij}^s = \mathbf{f}_i \qquad (1)$$

where node i has mass m_i, position $\mathbf{x}_i(t) = [x_i(t), y_i(t), z_i(t)]$, velocity $\dot{\mathbf{x}}$, and damping factor γ_i, and where $\mathbf{f}_i$ is an external force. Spring S_{ij}, which connects node i to neighboring nodes $j \in N_i$, exerts the force $\mathbf{f}_{ij}^s(t) = -(c_{ij}e_{ij} + \gamma_{ij}\dot{e}_{ij})\mathbf{r}_{ij}/||\mathbf{r}_{ij}||$ on node i (and it exerts the force $-\mathbf{f}_{ij}^s$ on node j), where c_{ij} is the elastic constant, γ_{ij} is the damping constant, and $e_{ij}(t) = ||\mathbf{r}_{ij}|| - l_{ij}$ is the deformation of the spring with separation vector $\mathbf{r}_{ij}(t) = \mathbf{x}_j - \mathbf{x}_i$. The natural length of the spring is l_{ij}.

Some of the springs in the biomechanical model play the role of contractile *muscles*. Muscles contract as their natural length l_{ij} decreases under the autonomous control of the motor center of the artificial animal's brain [17]. To dynamically contract a muscle, the brain must supply an *activation function* $a(t)$ to the muscle. This continuous time function has range $[0, 1]$, with 0 corresponding to a fully relaxed muscle of length l_{ij}^r and 1 to a fully contracted muscle of length l_{ij}^c. More specifically, for a muscle spring, $l_{ij} = al_{ij}^c + (1-a)l_{ij}^r$.

Typically, individual muscles form muscle groups, called *actuators*, that are activated in unison. Referring to Fig. 1 for example, the 40 muscles in the snake model are grouped pairwise in each segment to form 10 left actuators and 10 right actuators. Each actuator i is activated by a scalar *actuation function* $u_i(t)$, whose range is again normalized to $[0, 1]$. The actuation function transforms straightforwardly into activation functions for each muscle in the actuator. Thus, to control the snake's body we must specify the actuation functions $\mathbf{u}(t) = [u_1(t), \ldots, u_i(t), \ldots, u_N(t)]'$, where $N = 20$.

The continuous vector-valued function of time $\mathbf{u}(t)$ is called the *controller* and its job is to produce locomotion. Controllers may be stored within the artificial animal's motor control center.

2.2 Objective Functional

A continuous *objective functional* E provides a quantitative measure of the progress of the locomotion learning process. The functional is the weighted sum of a term E_u that evaluates the controller $\mathbf{u}(t)$ and a term E_v that evaluates the motion $\mathbf{v}(t)$ that the controller produces in a time interval $t_0 \leq t \leq t_1$, with smaller values of E indicating better controllers $\mathbf{u}$. Mathematically,

$$E(\mathbf{u}(t)) = \int_{t_0}^{t_1} \big(\mu_1 E_u(\mathbf{u}(t)) + \mu_2 E_v(\mathbf{v}(t))\big)\, dt, \qquad (2)$$

where μ_1 and μ_2 are scalar weights. Fig. 2 illustrates this schematically.

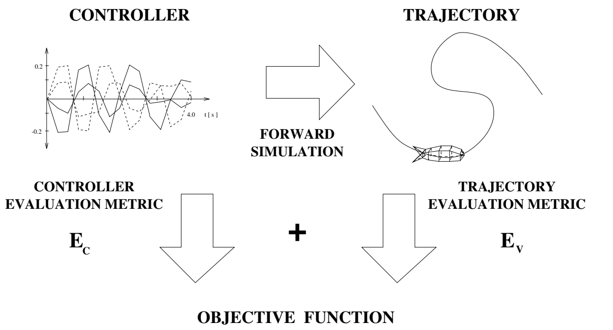


Figure 2: *The objective function guiding the optimization is a weighted sum of terms that evaluate the trajectory and the control function.*

It is important to note that the complexity of our models precludes the closed-form evaluation of E. As the figure indicates, to compute E, the artificial animal must first engage $\mathbf{u}(t)$ to produce a motion $\mathbf{v}(t)$ with its body (in order to evaluate term E_v). This is done through forward simulation of the biomechanical model over the time interval $t_0 \leq t \leq t_1$ using the controller $\mathbf{u}(t)$.

We may want to promote a preference for controllers with certain qualities via the controller evaluation term E_u. For example, we can guide the optimization of E by discouraging large, rapid fluctuations of $\mathbf{u}$, since chaotic actuations are usually energy inefficient. We encourage lower amplitude, smoother controllers through the function

$$E_u = \frac{1}{2}\left(\nu_1 \left|\frac{d\mathbf{u}}{dt}\right|^2 + \nu_2 \left|\frac{d^2\mathbf{u}}{dt^2}\right|^2\right), \qquad (3)$$

with weighting factors ν_1 and ν_2. The two component terms in (3) are potential energy densities of linear and cubic variational splines in time, respectively. The former penalizes actuation amplitudes, while the latter penalizes actuation variation.

The distinction between good and bad control functions also depends on the goals that the animal must accomplish. In our learning experiments we used trajectory criteria E_v such as the final distance to the goal, the deviation from a desired speed, etc. These and other criteria will be discussed shortly in conjunction with specific experiments.

2.3 Time and Frequency Domain Discrete Controllers

To solve the low level control problem, we must optimize the objective functional (2). This cannot be done analytically. We convert this continuous optimal control problem to an algebraic parameter optimization problem [5] by parameterizing the controller through discretization using basis functions. Mathematically, we express

$$u_i(t) = \sum_{j=1}^{M} u_i^j B^j(t), \qquad (4)$$

where the u_i^j are scalar parameters and the $B^j(t)$, $1 \leq j \leq M$ are (vector-valued) temporal basis functions. There are two qualitatively different choices of basis functions—local and global.

In the local discretization, the parameters u_i^j are nodal variables and the $B^j(t)$ can be spline basis functions. The simplest case is when the u_i^j are evenly distributed in the time interval and the $B^j(t)$ are tent functions centered on the nodes with support extending to nearest neighbor nodes, so that $\mathbf{u}(t)$ is the linear interpolation of the nodal variables (Fig. 3, top). Smoother B-splines can be used in a similar fashion. Since the nodal parameters are naturally ordered in a time sequence, we will refer to locally discretized controllers as *time domain controllers*.

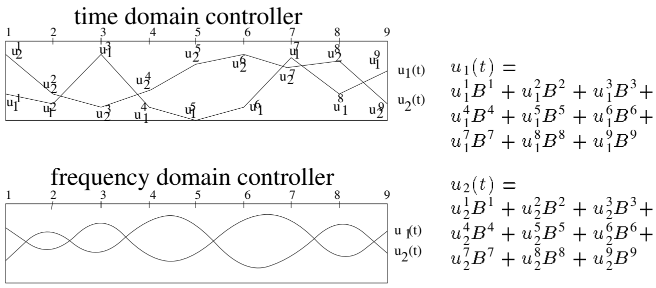


Figure 3: *Simple time domain controller (top) with two control functions $u_1(t)$ and $u_2(t)$. Each function is a piecewise linear polynomial generated by 9 control points. Simple frequency domain controller (bottom) with two control functions, each a sum of 9 sinusoidal basis functions $B^j(t) = \cos(\omega^j t + \phi^j)$.*

In the global discretization, the support of the $B^j(t)$ covers the entire temporal domain $t_0 \leq t \leq t_1$. A standard choice is sinusoidal basis functions $B^j(t) = \cos(\omega^j t + \phi^j)$ where ω^j is the angular frequency and ϕ^j is the phase, and the parameters u_i^j are amplitudes (Fig. 3, bottom). We will refer to controllers discretized globally using sinusoidal bases of different frequencies and phases as *frequency domain controllers*.

The time domain and frequency domain representations offer different benefits and drawbacks. The time domain controller yields a faster low-level learning rate. This issue is discussed in detail in Section 4.1. The frequency domain controller, on the other hand, does not require a change of basis during the abstraction process described in Section 3. It can also sometimes be extended arbitrarily in time since it favors periodicity.

2.4 Optimization of the Discrete Objective Function

Since $\mathbf{u}(t)$ has N basis functions, the discretized controller is represented using NM parameters. Substituting (4), into the continuous objective functional (2), we approximate it by the discrete *objective function* $E([u_1^1, \ldots, u_N^M]')$.

Learning low level control amounts to using an optimization algorithm to iteratively update the parameters of a time domain or frequency domain controller so as to maximize the discrete objective function and produce increasingly better locomotion. We have used both simulated annealing and the simplex method to optimize the objective function. The reader should refer to a text such as [12] for details about these optimization methods.

Simulated annealing has three features that make it particularly suitable for our application. First, it is applicable to problems with a large number of variables yielding search spaces large enough to make exhaustive search prohibitive. Second, it does not require gradient information about the objective function. Analytic gradients are not directly attainable in our situation since evaluating E requires a forward dynamic simulation of the animal. Third, it avoids getting trapped in local suboptima of E. In fact, given a sufficiently slow annealing schedule, it will find a global optimum of the objective functional. Robustness against local suboptima can be important in obtaining control functions that produce realistic motion. The benefit of using the simplex method over simulated annealing in some cases is its faster convergence rate. On the other hand, since it is a local optimization technique, strictly speaking, it can be applied successfully only to the class of optimization problems in which the topography of E is globally convex. Section 4.1 will describe in more detail the advantages and pitfalls of both methods when applied to the low level learning problem.

All of the biomechanical models described in Appendix A have demonstrated the ability to learn effective low level time domain locomotion controllers. Plates 1, 2, and 3 show frames from animations with controllers that have been learned by the snake, ray, and shark models, which produce natural and effective locomotion. Plate 3 illustrates a race among four sharks that have learned for different durations. The shark that is furthest from the camera has learned how to locomote for the shortest period of time, which yields muscle control functions that are essentially random and achieve negligible locomotion. Sharks closer to the camera have learned for progressively longer periods of time. The closest shark, which locomotes the best wins the race.

3 Abstracting High Level Control

It is time consuming to learn a good solution for a low level controller because of the high dimensionality of the problem (large NM), the lack of gradient information to accelerate the optimization of the objective functional, and the presence of suboptimal traps that must be avoided. Consequently, it is costly to produce animation by perpetually generating new controllers. The learning procedure must be able to abstract compact higher level controllers from the low level controllers that have been learned, retain the abstracted controllers, and apply them to future locomotion tasks.

The process of abstraction takes the form of a dimensionality reducing change of representation. More specifically, it seeks to compress the many parameters of the discrete controllers to a compact form in terms of a handful of basis functions. Natural, steady-state locomotion patterns tend to be quasi-periodic and they can be abstracted very effectively without substantial loss. The natural choice, therefore, is to represent abstracted controllers using the global sinusoidal basis functions discussed earlier. For the frequency domain controller, the dimensionality reduction is achieved trivially by retaining all basis functions whose amplitudes u_i^j exceed a low threshold and suppressing those below threshold. This results in a small set of significant basis functions with associated amplitudes that constitute the abstracted controller. To abstract a time domain controller, we apply the fast Fourier transform (FFT) [12] to the parameters of the time domain controller and then suppress the below-threshold amplitudes.

3.1 Using Abstracted Controllers

Typically, our artificial animals are put through a "basic training" regimen of primitive motor tasks that it must learn, such as locomoting at different speeds and executing turns of different radii. They learn effective low level controllers for each task and retain compact representations of these controllers through controller abstraction. The animals subsequently put the abstractions that they have learned into practice to accomplish higher level tasks, such as target tracking or leaping through the air. To this end, abstracted controllers are concatenated in sequence, with each controller slightly overlapping the next. To eliminate discontinuities, temporally adjacent controllers are smoothly blended together by linearly fading and summing them over a small, fixed region of overlap, approximately 5% of each controller (Fig. 4).

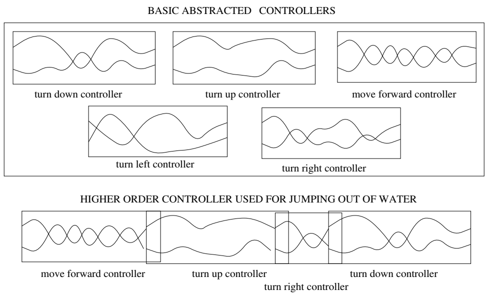


Figure 4: *Higher level controller for jumping out of water is constructed from a set of abstracted basic controllers.*

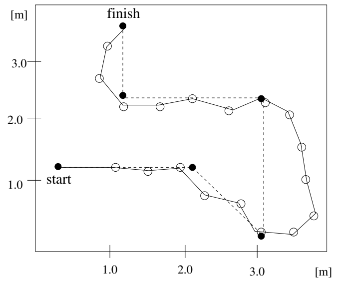


Figure 5: *The solid curve indicates the path of the fish tracking the goal. Black dots mark consecutive positions of the goal and white dots mark the starting point of a new controller.*

Currently we use abstracted controllers in two ways. In one scenario, the animated model has learned a repertoire of abstracted controllers that it applies in a greedy fashion to navigate in the direction of a target. It modifies its locomotion strategy periodically by invoking the abstracted controller that gives it the greatest immediate gain over the subsequent time interval. For example, Fig. 5 shows a path generated by the shark model using this method as it is presented with a series of targets. The shark has learned six basic abstracted controllers to accomplish this task: do-nothing, go-straight, sharp-turn-left, sharp-turn-right, wide-turn-left, and wide-turn-right. It then discovered how to sequence these controllers, and for what durations to apply them in order to locomote to successive targets indicated by black dots. The circles on the path in Fig. 5 indicate positions where the shark reevaluated its current strategy. Changes in the path's direction indicate that the shark has switched to a different controller which provides a bigger immediate gain. Plate 4 shows rendered frames from the locomotion animation with the targets rendered as red buoys. This method is inexpensive and can be made to work in real time. Unfortunately, the greedy strategy is bound to fail on a problem that requires careful planning.

Plate 1: *Locomotion pattern learned by the artificial snake.*

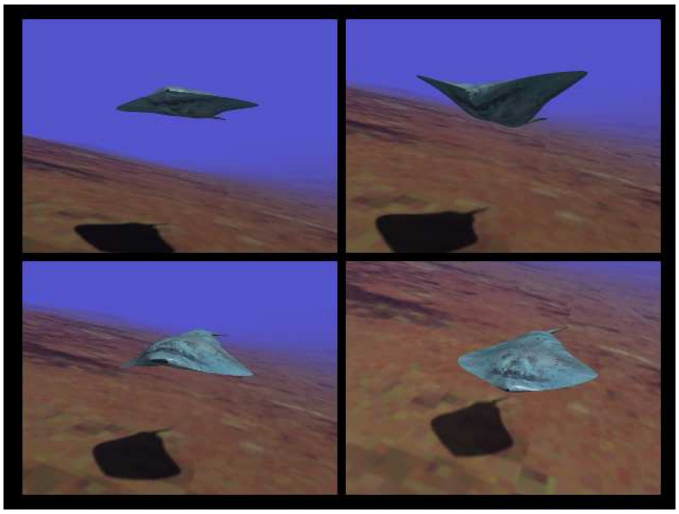


Plate 2: *Locomotion pattern learned by the artificial ray.*

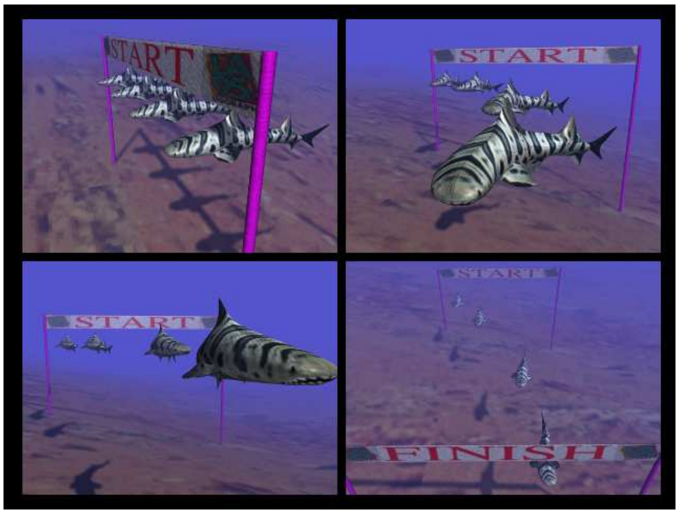


Plate 3: *Shark race illustrates the progress of learning (see text).*

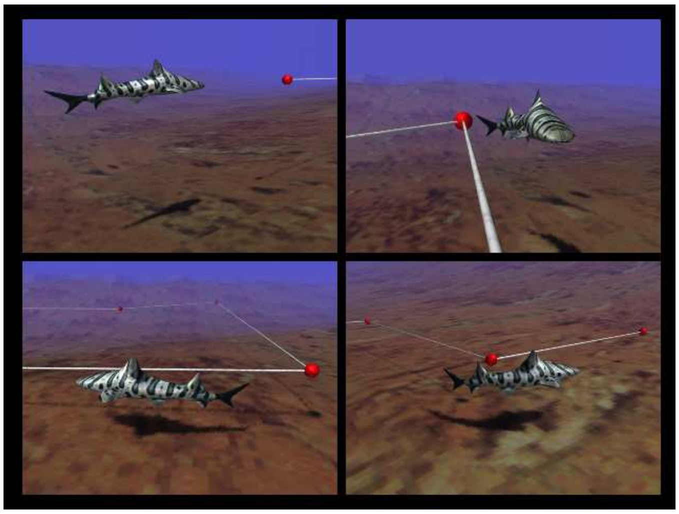


Plate 4: *Target tracking using abstracted controllers (see text).*

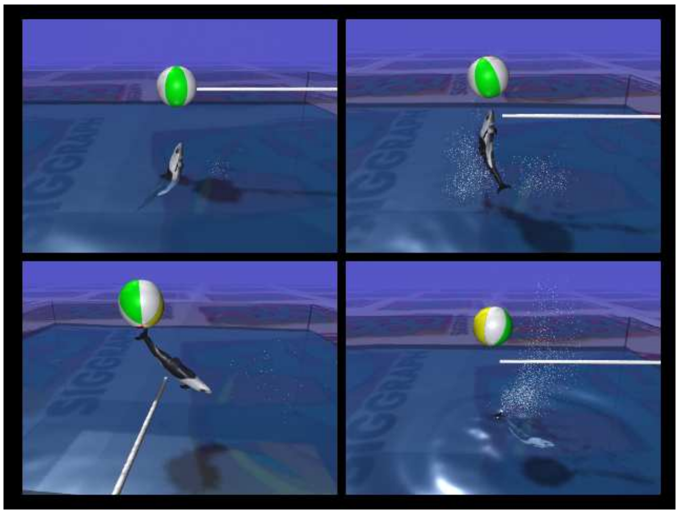


Plate 5: *SeaWorld tricks learned by the artificial dolphin.*

The second method overcomes the limitations of the greedy strategy by learning composite abstracted controllers that accomplish complex locomotion tasks. Consider the spectacular stunts performed by marine mammals that elicit applause at theme parks like “SeaWorld”. We can treat a leap through the air as a complex task that can be achieved using simpler tasks; e.g., diving deep beneath a suitable leap point, surfacing vigorously to gain momentum, maintaining balance during the ballistic flight through the air, and splashing down dramatically with a belly flop.

We have developed an automatic learning technique that constructs a macro jump controller of this sort as an optimized sequence of basic abstracted controllers. The optimization process is, in principle, similar to the one in low level learning. It uses simulated annealing for optimization, but rather than optimizing over nodal parameters or frequency parameters, it optimizes over the selection, ordering, and duration of abstracted controllers. Thus the animal model applying this method learns effective macro controllers of the type shown at the bottom of Fig. 4 by optimizing over a learned repertoire of basic abstracted controllers illustrated at the top of the figure.

3.2 Composing Macro Controllers

We first train the artificial dolphin so that it learns controllers for 5 basic motor tasks: turn-down, turn-up, turn-left, turn-right, and move-forward. We then give it the task of performing a stunt like the one described above and the dolphin discovers a combination of controllers that accomplishes the stunt. In particular, it discovers that it must build up momentum by thrusting from deep in the virtual pool of water up towards the surface and exploit this momentum to leap out of the water. Plate 5(a) shows a frame as the dolphin exits the water. The dolphin can also learn to perform tricks while in the air. Plate 5(b) shows it using its nose to bounce a large beach-ball off a support. The dolphin can learn to control the angular momentum of its body while exiting the water and while in ballistic flight so that it can perform aerial spins and somersaults. Plate 5(c) shows it in the midst of a somersault in which it has just bounced the ball with its tail instead of its nose. Plate 5(d) shows the dolphin right after splashdown. In this instance it has made a dramatic bellyflop splash. By discovering controllers that enable it to control its body in these complex ways, the dolphin can amuse and entertain the animator, who would be hard pressed to design similar controllers by hand for a physics-based model with as many control parameters as the dolphin model has.

To train the dolphin to perform a variety of stunts we introduced additional "style" terms into the objective function that afford extra control on the animal's trajectory in the air. For example, a simple jump was learned by optimizing over the maximum height at some point in time. In order to train the dolphin to jump a hurdle, we introduced a term to control its orientation as it clears the hurdle—at the apex of its trajectory, it should be in a horizontal orientation and it should face downward upon reentry. The somersault controller was discovered by adding a term that encouraged maximum angular velocity of the body in flight. We can maximize or minimize the area of the animal's body that hits the water upon reentry for crowd-drenching splashes or for high scores from the judges for a clean dive.

Different style terms can, in principle, be added indefinitely to the control function. The only limitation seems to be the increasing complexity of the optimization. We have noted that although it can produce some interesting results, simulated annealing is not especially well suited to this kind of optimization problem. We suspect that this is due to the combinatorial nature of the problem and the fact that simulated annealing does not take advantage of partial solutions that it finds along the way, but instead starts with a new set of parameters at every iteration of the optimization process. Unfortunately, at a high level of abstraction sometimes even small changes in the set of parameters produce drastically different trajectories. Genetic algorithms may perform better on such problems.

4 Additional Experiments and Results

This section presents a more thorough experimental study of our approach and reports additional results.

4.1 Performance of the Optimizers

Fig. 6(a) compares the performance of the simplex method and simulated annealing in seeking optimal time and frequency domain controllers for the shark model given the task of locomoting to a specified goal location; i.e., $E_v = ||\mathbf{d}_g||^2$, where $\mathbf{d}_g$ is the separation vector between the nose node of the shark and the goal point. The term E_u (3) in the objective functional (2) was disabled for these tests ($\mu_2 = 0$). For simplicity, the 4 muscles in each segment were grouped into a single actuator ($N = 3$ actuators total), with the left muscle pair receiving contraction signals that are exactly out of phase with the right muscle pair. A time interval was discretized using $M = 15$ parameters; hence, the dimensionality of the search space was $NM = 45$.

Both methods converge to good time domain controllers. The simplex method yields a final objective functional value of $E_o = 0.49$ after approximately 500 iterations. Simulated annealing finds a slightly better solution, $E_o = 0.42$, but only after 3500 iterations. For frequency domain controllers, the results differ substantially. Simulated annealing performs almost as well as for the time domain controller, yielding an objective function value of $E_o = 0.52$ after 3500 iterations.[2] However, the simplex method does much worse—it fails to get below $E_o = 0.65$.

Fig. 6(b–c) compares the convergence for simulated annealing on both types of controllers. The results are better for the objective function represented in the time domain (Fig. 6(b)) than for the frequency domain representation (Fig. 6(c)). For the time domain representation we need approximately 700 iterations to get very close to the global minimum. The number of iterations for the frequency domain representation is much greater.

The above results suggest that it is much harder to optimize the objective using frequency domain controllers. To understand why, we plotted E against pairs of randomly chosen parameters u_i^j (labeled x and y in the plots), using both time and frequency domain representations. We stepped the selected parameters through a range of values while keeping the other parameters constant and evaluated the objective function repeatedly to obtain a 3D surface plot (each repetition required a forward simulation of the locomotion). The plot in Fig. 7(a) reveals the simple convex topography of E for time domain controllers, while the plot in Fig. 7(b) reveals the much more irregular, nonconvex topography of E for frequency domain controllers.

Evidently, small changes in the local basis functions of the time domain controller produce small, well-behaved changes in the value of the objective function. By contrast, small changes in the global basis functions of the frequency domain controller produce relatively larger, more irregular changes in the value of the objective function.

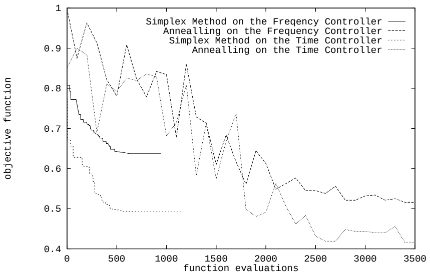
The many local minima in the topography of the objective function associated with the frequency domain controller lead to failure of the simplex method and they present more of a challenge to simulated annealing. The convex structure of the objective function associated with the time domain controller allows both annealing and simplex to converge very quickly. Moreover, they often yield better time domain controllers than frequency domain controllers. We conclude that the time domain controller representation is a worthwhile one.

4.2 Influencing Controllers via the Objective Functional

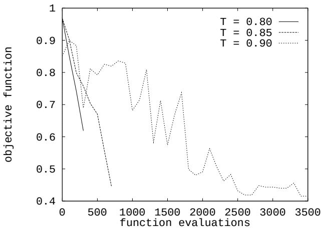
In Section 2.2 we discussed how the term E_u in (2) that evaluates controllers allows us to influence the motion. For example, we can discourage chaotic motions. This section investigates the effect of different E_u terms in more detail. Again, we employ the shark with the same goal E_v as in the previous section.

Fig. 8(a) shows the results obtained with the time domain representation for $\nu_2 = 0$, hence the discrete term $E_u = \nu_1/2h^2(u_i^{j+1} - u_i^j)^2$, where h is the timestep between nodal parameters. The objective function was evaluated for $\nu_1 = 0.0, 0.1, 0.2$ (top to bottom). As the value of ν_1 increases, both the amplitude and the frequency

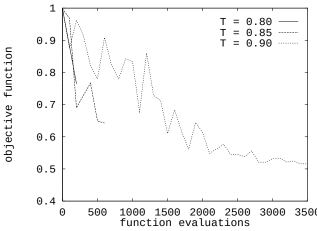
[2] We start the simulated annealing procedure at the temperature $T_0 = 0.5$. The annealing schedule is $T_{i+1} = \alpha T_i$ where $0 \leq \alpha \leq 1$. Usually $\alpha = 0.9$, but Fig. 6(b–c) shows the results obtained with different schedules. The maximum number of steps before the temperature drop is $10NM$ and the minimum number of accepted perturbations before the temperature drop is NM For $N = 15$ it takes about one hour on an SGI Indigo workstation to get a solution for each control function. At each step the values of all parameters are perturbed randomly. The perturbation is bounded by 10% of the range of admissible values. So, for example, if the maximum contraction of the muscle is 20% of its relaxed length, each perturbation will be at most 2%. The bound on the perturbation remains fixed over the annealing process. This yields much faster convergence than if we decreased the magnitude of the perturbation with temperature.



(a)

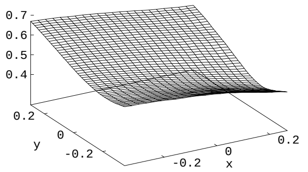


(b)

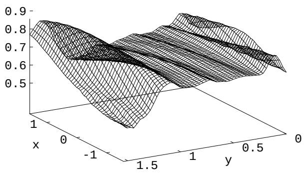


(c)

Figure 6: *(a) Performance comparison of the simplex method and of simulated annealing. Convergence rate of simulated annealing on the time domain controller (b) and on the frequency controller (c) with cooling rates:* $T_0 = 0.8$*,* $T_1 = 0.85$*, and* $T_2 = 0.9$*.*



(a)



(b)

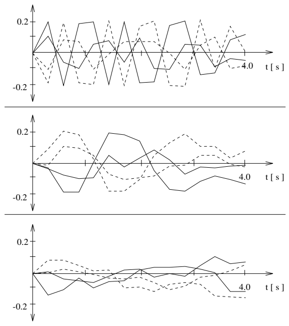
Figure 7: *Topography of objective function (in 2 dimensions) of time domain representation (a) and frequency domain representation (b).*

of the learned actuation functions drop and the shark learns locomotion controllers that result in less energy expenditure. Fig. 8(b) shows the results obtained with $\nu_1 = 0$, hence the discrete term $E_u = \nu_2/2h^4(u_i^{j+1} - 2u_i^j + u_i^{j-1})^2$ with $\nu_2 = 0.0, 0.002, 0.006$ (top to bottom). As the value of μ increases, the amplitude of the learned actuation functions remains constant and only the frequency decreases.

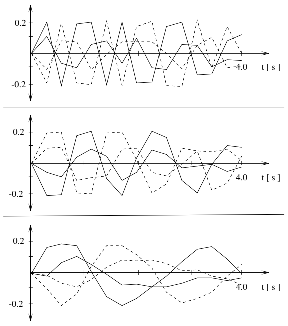
4.3 Abstracted Learning Results

Next, we report results for the shark and the snake in abstracting learned time domain controllers for straight locomotion and left turns. Right turn controllers were obtained by swapping signals sent to left and right actuators.

To obtain good abstracted controllers for the shark, it was sufficient for both straight motion and left turn to retain the single dominant mode of the FFT. Fig. 9(a) shows learned controllers for the shark swimming straight. In this experiment, the left and right muscle pairs in each segment constitute independent actuators, but note how the animal has learned to actuate its left muscles approximately out of phase with those on the right side. The posterior segment muscles contract with roughly half the frequency of the other muscles and the muscles on either side of the body are activated in sequence with a slight phase shift. For the swim left



(a)



(b)

Figure 8: *Influence of* E_u *on controller: (a)* $\nu_2 = 0$*; (b)* $\nu_1 = 0$*.*

sequence (Fig. 9(b)) the posterior and anterior segment muscles on the right side of the body are essentially unused, while all the muscles on the left side of the body move approximately in phase.

It suffices to use two primary modes of the Fourier transform to get an effective abstracted controller for the turning snake. For straight locomotion (Fig. 9(c)) it is difficult to interpret the time domain actuation functions. However, if we look at the learned abstracted controller for straight locomotion, we can clearly see that the main modes have the same frequency, but their phase is shifted slightly, as expected. When a real snake turns, it first coils itself in the direction of the turn then switches back to its normal serpentine mode of locomotion. This pattern is revealed in the automatically learned turn controller shown in Fig. 9(d). First, all the muscles on the right side of the body relax and all the muscles on the left side contract. Then they resume their action for straight serpentine locomotion.

A Biomechanical Model Structure and Simulation

Animals such as snakes, worms, fishes, and marine mammals, with highly flexible bodies are well suited to mechanical modeling using spring-mass systems. All of our animal body models consist of an internal biomechanical model with contractile muscles coupled to an external, texture-mapped, NURBS surface display model.

Fig. 1 shows the spring-mass system for the coral snake (*Micruroides euryxanthus*), which is similar to the one in [8]. Plate 1 shows the display model. The muscle springs can contract to 30% of their relaxed length. The body mass is distributed evenly among all nodes.

Fig. 10 shows the spring-mass system for the Leopard shark (*Triakis semifasciata*) [2], which is similar to the fish model in [17]. Plates 3 and 4 show the display model. The 4 posterior muscles can contract to 10% of their relaxed length; the 8 other muscles to 20%. The figure specifies the nodal mass distribution.

We model a Heaviside's dolphin (*Cephalorhynchus heavisidii*) [11] (Plate 5 shows the display model) straightforwardly by turning the shark spring-mass system on its side, such that the muscles serve as caudal (tail) fin elevator and depressors. We equip the dolphin with functional pectoral fins that allow it to roll, pitch, and yaw in the water (see [17] for details about fins).

Fig. 11 shows the spring-mass system for the Kuhl's stingray (*Dasyatis kuhlii*) [13]. Plate 2 shows the display model. Four left and 4 right elevator muscles and an equal number of depressor muscles are capable of flexing the wings by contracting to 20% of their relaxed length. Mass is distributed evenly among all the nodes.

To model snake locomotion, we use directional friction against the ground which generates reaction forces that move the body forward, as described in [8]. To model marine animal locomotion, we compute hydrodynamic reaction forces acting on each of the

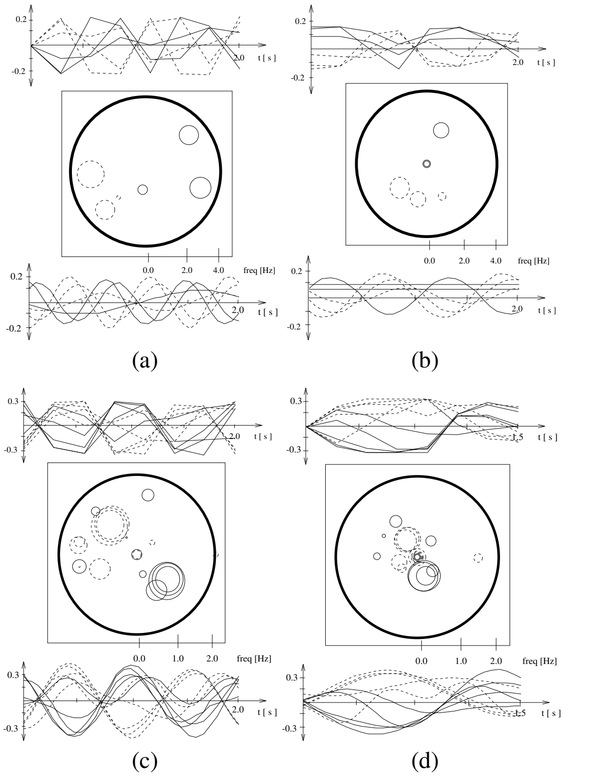


Figure 9: *Learned controller for the swim straight (a) and the left turn (b) for the shark. Learned controller for the straight motion (c) and the left turn (d) for the snake. For each part: (top) learned time domain controller (dotted lines indicate actuator functions for left side of body, solid lines indicate actuator functions for right side); (center) primary modes of controller FFT (radius of circles indicates mode amplitudes, radial distances from center of surrounding circle indicate frequencies, angular positions within surrounding circle indicate phases); (bottom) abstracted controller obtained by retaining primary modes.*

model's faces, as described in [17]. These forces produce external nodal forces $\mathbf{f}_i$ in the equations of motion (1).

We use a stable, efficient semi-implicit Euler method [12] to numerically integrate these ODEs. It is implicit in the internal forces on the lhs of (1) and explicit in the external forces $\mathbf{f}_i$.

Acknowledgments

We thank Xiaoyuan Tu, who developed the original fish biomechanical model, for her software and cooperation, and Geoffrey Hinton for valuable discussions. This work was made possible by a grant from the Natural Sciences and Engineering Research Council of Canada. DT is a fellow of the Canadian Institute for Advanced Research.

References

[1] L. S. Brotman and A. N. Netravali. Motion interpolation by optimal control. Proc. ACM SIGGRAPH, 22(4):309–407, 1988.

[2] J. I. Castro. *The Sharks of North American Waters*. Texas Unviersity Press, 1983.

[3] M. F. Cohen. Interactive spacetime control for animation. Proc. ACM SIGGRAPH, 26(2):293–301, 1992.

[4] C. H. Ernst. *Venomous Reptiles of North America*. Smithsonian Institution Press, 1992.

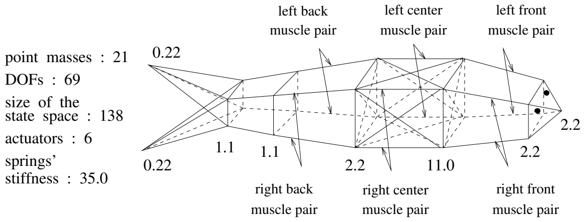


Figure 10: *The shark biomechanical model has six actuators consisting of a pair of muscles that share the same activation function. The numbers along the body indicate the mass of each point in the corresponding cross sectional plane. The cross-springs that maintain the structural integrity of the body are indicated in one of the segments only.*

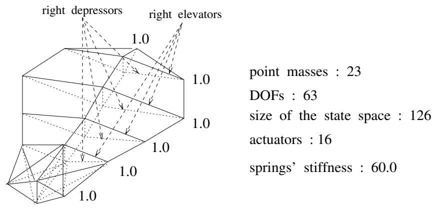


Figure 11: *The ray biomechanical model has four sets of actuators: left and right depressors and left and right elevators. The numbers along the body indicate the mass of each point in the corresponding cross sectional plane. The cross-springs that maintain the structural integrity of the body are indicated in one of the segments only.*

[5] C. J. Goh and K. L. Teo. Control parameterization: A unified approach to optimal control problems with general constraints. *Automatica*, 24:3–18, 1988.

[6] R. Grzeszczuk. Automated learning of muscle based locomotion through control abstraction. Master's thesis, Dept. of Comp. Sci., Univ. of Toronto, Toronto, ON, January 1994.

[7] R. McNeill Alexander. *Exploring Biomechanics: Animals in Motion*. Scientific American Library, New York, 1992.

[8] G.S.P. Miller. The motion dynamics of snakes and worms. Proc. ACM SIGGRAPH, 22(4):169–177, 1988.

[9] J. T. Ngo and J. Marks. Spacetime constraints revisited. Proc. ACM SIGGRAPH, 27(2):343–351, 1993.

[10] M. G. Pandy, F. C. Anderson, and D. G. Hull. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *Transactions of the ASME*, 114(450), November 1992.

[11] Best P.B. and Abernethy R. B. Heaviside's dolphin. In *Handbook of Marine Mammals*, volume 5, pages 289–310. Academic Press, 1994.

[12] W. H. Press, B. Flannery, et al. *Numerical Recipes: The Art of Scientific Computing, Second Edition*. Cambridge University Press, 1992.

[13] J. E. Rendal, Allen G. R., and Steene R. C. *Fishes of the Great Barrier Reef and Coral Sea*. Univ. of Hawaii Press, Honolulu, HI, 1990.

[14] G. Risdale. Connectionist modeling of skill dynamics. *Journal of Visualization and Computer Animation*, 1(2):66–72, 1990.

[15] K. Sims. Evolving virtual creatures. Proc. ACM SIGGRAPH, pages 15–22, 1994.

[16] 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.

[17] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. Proc. ACM SIGGRAPH, pages 43–50, 1994.

[18] M. van de Panne and E. Fiume. Sensor-actuator networks. Proc. ACM SIGGRAPH, 27(2):335–343, 1993.

[19] A. Witkin and M. Kass. Spacetime constraints. Proc. ACM SIGGRAPH, 22(4):159–167, 1988.