Sensor-Actuator Networks

Michiel van de Panne
*Eugene Fiume**

Department of Electrical Engineering and *Computer Science
University of Toronto†
Toronto, Canada, M5S 1A4

Abstract

Sensor-actuator networks (SANs) are a new approach for the physically-based animation of objects. The user supplies the configuration of a mechanical system that has been augmented with simple sensors and actuators. It is then possible to automatically discover many possible modes of locomotion for the given object. The SANs providing the control for these modes of locomotion are simple in structure and produce robust control. A SAN consists of a small non-linear network of weighted connections between sensors and actuators. A stochastic procedure for finding and then improving suitable SANs is given. Ten different creatures controlled by this method are presented.

CR Categories: G.3 [Probability and Statistics]: Probabilistic Algorithms; I.2.6 [Artificial Intelligence]: Learning, Robotics; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - animation; I.6.3 [Simulation and Modeling] - Applications.

1.0 Introduction

Many recent investigations have involved the use of physical simulations for creating life-like animations of creatures. The problem is less one of simulation than it is of control: how should the muscles in the creature act to make it move in a desired way? Once known, the forces and torques produced by the muscles can be used as input to a physical simulation to obtain the resulting motion.

Broadly speaking, there exist two approaches to this problem. The first is to apply the laws of physics as constraints on the motion of the various parts of the creature while at the same time optimizing a given goal function. This goal function may be something like "get from point A to point B in the fastest way". An initial guess at the possible trajectory of the creature is iteratively refined until a trajectory is produced that both satisfies the physics constraints and optimizes the goal function.

The second approach is to synthesize a controller. A controller makes control decisions based upon sensory information presently available and does not explicitly calculate a trajectory. A controller thus makes use of feedback to perform its task. Our solutions, called sensor-actuator networks (SANs) are in the form of controllers. There are two main novel features of SAN controllers. First, they are entirely sensor-based. Control solutions are usually cast into a form where sensory information is processed to produce an estimate of the system state. For mechanical systems, the state is the tuple of values sufficient to specify the position and velocity of every point on the object. SANs have no notion of the state of the system as conventionally defined. Second, the stochastic synthesis procedures used to create SANs are unique and powerful tools.

A principal advantage of our method is that it requires less knowledge to use than other physically-based animation systems. The user provides the construction of a creature and can then ask the question "how would it move?" Our method can then yield several controllers providing feasible modes of locomotion (there is no *guarantee* that a suitable controller will be found). We have been surprised with many of the modes of locomotion discovered. Even the simplest objects are capable of a large repertoire of interesting motions. Many of the controllers generated using our method can take advantage of phenomena such as surface friction or collision impact, which can be simulated but are hard to incorporate in many optimization methods. Figure 1 shows modes of locomotion discovered for 4 different creatures.

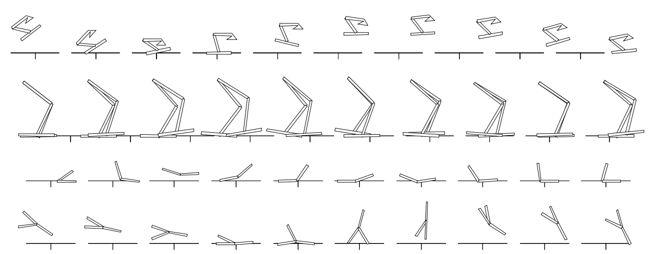


FIGURE 1. Some modes of locomotion using SANs

A sensor-actuator network is a non-linear network of weighted connections between a small number of binary sensors and the actuators (the muscles of our creatures). The network has internal delays, thereby giving it dynamic properties. In this paper we provide a method for determining the parameters associated with SANs in order to obtain instances of a desired behaviour. We show that SANs can be used to control a variety of interesting creatures.

The next section relates our work to important previous work. Section 3 discusses how to construct a creature. Section 4 presents the architecture of SANs. Section 5 gives the algorithms for synthesizing and fine-tuning SANs. Section 6 presents a variety of results, and section 7 concludes.

† Email: van@dgp.utoronto.ca, elf@dgp.utoronto.ca

The financial assistance of the Natural Sciences and Engineering Research Council of Canada, and of the Information Technology Research Centre of Ontario, is gratefully acknowledged.

2.0 Background

Developing control solutions for arbitrary dynamical systems is a difficult problem. Consequently, control solutions are usually specific to a constrained class of control problems. We briefly consider various classes of control problems applicable to animation and their solution techniques.

Linear systems are often the simplest to control. A linear system can be written in the form $\dot{x} = Ax + b$ where x is the state vector. Brotman and Netravali[5] applied linear control theory to the animation of some linear systems. Unfortunately, most interesting systems are very non-linear.

We shall define *smooth* systems as being those that are not necessarily linear, but whose state variables are C^0 continuous over time. This implies positions and velocities are C^1 and C^0 continuous, respectively. This notably excludes mechanical systems that undergo collisions, which instantaneously lose velocity. Smooth systems have states that follow a continuous trajectory through state-space. The solution to smooth control problems typically involves iteratively refining a state trajectory[6][17][24] or performing dynamic programming[8][22]). Solutions to smooth problems are often useful for animation. The take-off and aerial trajectories of many jumping motions fall within this class.

Statically stable systems are those that can be effectively controlled through kinematic means. This means that the motion of a creature can be halted in mid-action and then resumed without it falling over or otherwise collapsing. These can often be controlled by a cyclic motion passing through the body or legs. Realistic animations of snakes and worms[14] and a cockroach[13] have been performed using suitable controllers.

Specialized controllers have been constructed for many systems through careful analysis and simplification of the motions involved. Examples of interesting specialized controllers for walking, hopping, swinging, and juggling are given in [9][18][20][21]. The work in *Making Them Move*[1] presents a variety of specific solutions to control problems in animation.

We shall deal with a class of control problems in which the systems are non-linear, non-smooth, and not statically stable. A large number of crawling, jumping, hopping, flipping, and walking creatures fall in this category. Our sensor-actuator networks (SANs) prove to be particularly adept at controlling creatures in this class. It is difficult to try to perform an optimization that incorporates discontinuities. It should be noted, however, that it is not difficult to simulate many systems of this class.We shall use this to our advantage by performing repeated trials to determine a suitable controller. This means that a minimal knowledge of the physics of the mechanical system is required. We carry out our trials using physical simulations, although they could equally well be carried out directly on the real objects in principle.

Our work has similarities with the work of Wilhelms and Skinner[23] and Braitenberg[3]. Both discuss controllers constructed using weighted connections between sensors and actuators (or *effectors*), possibly through intermediate nodes of some kind. Braitenberg presents a series of thought experiments showing that such networks are capable of producing complex and seemingly intelligent behaviour. Wilhelms and Skinner allow the user to interactively construct the mapping between sensors and actuators. Their creature consists of a rigid body that can propel itself in three dimensions using a jet. Several different kinds of nodes are suggested for use in the connection network. Examples of attraction and avoidance behaviours can be constructed using their approach. Our work will show that SANs, which are similar networks, can be *automatically* synthesized and can be used to control a large variety of physically realizable creatures. Furthermore, we show it is only necessary to consider *one* kind of network node.

Our synthesis technique embodies a *generate-and-test* philosophy that can also be found in some other recent work. Ngo and Marks[16] use it to achieve similar goals to ours. Maes and Brooks[11] learn to coordinate leg motions for an insect robot. Sims uses the same notion to produce striking images[19]. A discussion of search-based control methodology is given in [15].

McGeer's work on passive walking[12] illustrates the phenomenon of mechanical systems reaching and maintaining stable limit cycles. Such *attractors* or limit cycles have also been studied in neural networks and have been conjectured as being the basis for behavioural action in nervous systems. In our work, the dynamical system consisting of the SAN and the creature has a propensity for such limit cycles, in which case it results in periodic motions, some of which are the useful gaits we are looking for.

SANs are similar in topology to many artificial neural networks (ANNs), but are different in several respects. Non-recurrent ANNs have no internal delays and thus provide a static input-output mapping. Furthermore, the synthesis method we employ does not use derivative-based learning methods.

SANs are loosely related to Brooks' work[4] on subsumption architectures for the control of mobile robots. The subsumption architecture is used to implement control at various levels in a control hierarchy. Beer presents a model of an insect nervous system that produces interesting behaviour[2].

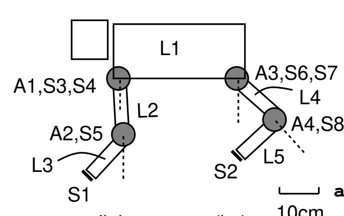
3.0 Constructing a Creature

The details of our method are best described through the use of an example. We shall use the *bounder* creature, shown in Figure 2. The bounder consists of 5 links, has 4 angular actuators, and 8 binary sensors. SANs take the binary sensor values as inputs and produce a set of desired angles or lengths as outputs. These desired values are used by proportional-derivative (PD) controllers to determine the internal torques or forces to apply. A PD controller is functionally equivalent to a spring and damper between the desired position and current position of a link. We shall now discuss each of the components of a creature in more detail and then discuss our simulator.

3.1 Mechanical Configuration

The user begins by specifying the desired mechanical configuration for the creature. All our creatures are built of rigid links and have planar dynamics, most of them operating in a vertical plane (i.e., under the influence of gravity). This is a limitation of our current implementation rather than the technique in general. As we shall

mechanical configuration:



link	mass (kg)
L1	1.0
L2	0.15
L3	0.10
L4	0.15
L5	0.10

sensors:

sensor	type	link	min	max
S1	touch			
S2	touch			
S3	angle	2	–180	–10
S4	angle	2	25	180
S5	angle	3	–180	–80
S6	angle	4	–180	42
S7	angle	4	55	180
S8	angle	5	–180	–105

actuators:

act.	min	max	ks	kd
A1	–20	30	0.4	0.01
A2	–85	–55	0.4	0.01
A3	40	55	0.4	0.01
A4	–110	–105	0.4	0.01

FIGURE 2. The bounder

soon see, planar creatures can have a very large repertoire of interesting behaviours. Specifying the physical structure of the creature requires specifying the mass, moment of inertia, and shape of each link, as well as how they are connected with joints.

3.2 Sensors

All sensors for our creatures are binary. If a sensor is 'on,' it produces a value of 1; otherwise it produces a 0. We currently use the four kinds of sensors shown in Figure 3. Touch sensors (e.g., S1 and S2 for the bounder) turn on when in contact with the ground and otherwise remain off. Angle sensors (e.g., S3-S8) determine if the angle of a limb is within the fixed range determined by the minimum and maximum angle specified for the sensor. This angle is measured relative to the link it is attached to, and the zero position is indicated with a dashed line in the creature diagrams. Eye sensors turn on if the *follow point* is in their cone of view. The follow point is a point to which some of our creatures will be attracted. One can thus control these creatures by dragging the follow point in front of them along a desired path. The cone of view for an eye is defined by a minimum and maximum angle. The zero-degree reference for eyes is shown in the creature diagrams with a dashed line. Length sensors are similar to angle sensors, but measure linear distances.

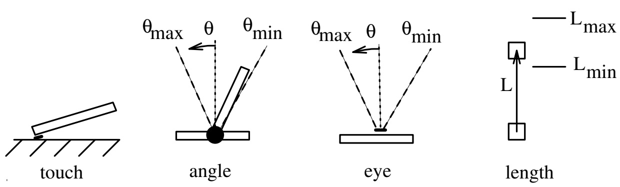


FIGURE 3. Sensor types

Sensors are added to a creature after the mechanical configuration has been designed. It is necessary to use some intuition and creativity in deciding what sensors will be useful to a creature. Touch sensors are useful for points expected to be in contact with the ground. Angle sensors are useful for detecting when a limb has reached the end of its swing. Eye sensors are necessary to be able to track a follow point. Length sensors are useful in the control of linear actuators. The total number of sensors should also be considered. Too many or too few sensors can introduce problems for the network synthesis technique to be outlined shortly.

3.3 Actuators

We shall use two kind of actuators in our creatures: *linear* and *angular.* These are shown in Figure 4. Specifying an angular actuator requires specifying the joint concerned, the upper and lower limits that the desired angle can take (measured in degrees in our creature diagrams), and specifying the *strength* of the actuator.

The strength of an actuator is determined by the constants k_s and k_d associated with the PD controller for the joint. An angular actuator generates an internal joint torque $T = k_s(\theta_d - \theta) - k_d\dot{\theta}$, while a linear actuator generates a force $F = k_s(l_d - l) - k_d\dot{l}$. For many motions, choosing $k_d / k_s = 0.10$ provides suitable energetic motions with some damping. The value of k_s can often be chosen by doing a simple calculation. For example, a 10 degree deflection of limb L5 of the bounder should give a large enough force to support half the body weight when the leg is bent. A simple calculation will produce $k_s = 0.4$ Nm/deg, which is used for all the actuators in the bounder. No hard joint limits are provided by the actuators, although they can of course be implemented in the simulaton if desired.

Length actuators are similar to angular actuators, except that they exert linear forces between a pair of points. Besides the two points, the minimum and maximum desired length allowable for

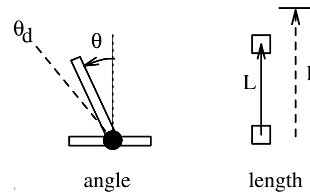


FIGURE 4. Actuator types

the actuator must be specified, as well as its strength. As with the angular actuators, the strength of the actuator is determined by its k_s and k_d constants.

3.4 Dynamics Simulation

Generating and using SANs requires only the capability to simulate the dynamics of a system. It is, however, beneficial to have a fast dynamics simulator, as our method for generating suitable SANs requires many simulation trials.

We make use of a dynamics compiler that uses the mechanical configuration information to generate a 'C' procedure which solves the equations of motion for a single time step. The procedure generates and solves a set of linear equations Ax = b, where x is the set of unknown accelerations. The elements in A and b are functions of the system state, the physical parameters of the system, the internal torques, and external forces. A recursive Newton-Euler formulation is used. This is $O(n^3)$ in the number of links and is quite suitable for $n < 10$.

The creatures are treated as free bodies in space. The external forces applied by the ground are calculated using stiff spring and dampers. We favour this approach as being simpler and more flexible than the alternative choice of reformulating the equations of motion upon impact of a link with the ground. The coefficient of friction of the ground can be set to a desired value. Other physical phenomena such as wind and water forces are easily added in as external forces. For example, for our fish creature, we calculate the water force for each link as $F = k\int(v \bullet N)\,dA$, where $v \bullet N$ defines the component of the velocity in the direction of the surface normal and A is the surface area of the link.

4.0 Sensor-Actuator Networks

SANs provide control by connecting sensors to actuators through a network of weighted connections. We will discuss in the next section the important question of how to determine the weight values. This section describes the structure and operation of SANs

A simplified example of a SAN is shown in Figure 5. The network consists of nodes and unidirectional weighted connections. The weights of the connections can take on values in a fixed range. In our implementation we choose integer values in the range [-2,2]. As shown, there are three kinds of nodes: sensor nodes, hidden nodes, and actuator nodes. The sensor nodes are fully connected to all hidden nodes and actuator nodes. All hidden nodes and actuator nodes are fully interconnected. The number of hidden nodes is usually chosen to be approximately equal to the number of sensor nodes. For example, the SANs for the bounder have 8 sensor nodes, 8 hidden nodes, and 4 actuator nodes.

Sensor nodes take on the values of their associated sensors. The hidden and actuator nodes function as shown in Figure 6. A node sums the weighted inputs and outputs a '1' if the sum is positive. This is a function similar to those performed in neural networks. It is important, however, for the controller to be a dynamical system on its own. This is effected by having a time delay associated with the operation of each node. This delay is implemented with the integrator and the following hysteresis function. The constants k_1

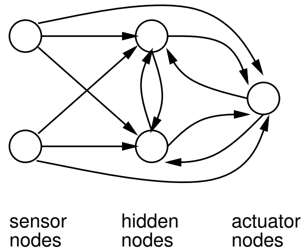


FIGURE 5. Topology of a sensor-actuator network

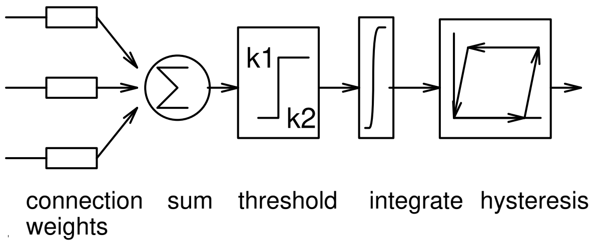


FIGURE 6. Function of a SAN node

```
sum = 0
for (each input i)
        sum = sum + input[i]*weight[i]
if (sum>0) then
        istate = istate + k1*dt
else
        istate = istate + k2*dt
if (istate>1.0) then istate = 1.0
if (istate<0.0) then istate = 0.0
if (output==1 and istate==0.0) then output=0
if (output==0 and istate==1.0) then output=1
```

FIGURE 7. Code corresponding to node function

and k_2 provide control over the time delay for a node to turn on and off, respectively. A hysteresis function is used instead of a simple threshold function to prevent the node output from chattering when some sensors make only momentary contact.

Actuator nodes make direct use of the result of the summing operation at a node to determine the desired angle or length of the actuator. The sum is first bounded to lie in a fixed range and is then linearly mapped to the length or angle range of the actuator. We bound the sum to the range [-2,2] in our implementation.

The function of a node is both simple to program and to build directly in hardware. The code implementing the functionality of a node is shown in Figure 7. The variable 'istate' is the internal state of a node associated with the integrator. We initialize istate to zero for all nodes at the start of a simulation. A hardware implementation would require three resistors, a capacitor, and three opamps per node.

The constants k_1 and k_2 should be chosen in relation to the expected duration of a periodic locomotion cycle for the creature. Thus the node delays for an elephant should be much larger than those for a mouse. The delays are given by $T_{on} = 1/k_1$ and $T_{off} = -1/k_2$ (note: $k_2 < 0$). Typically, good results can be obtained with $T_{on} = T_{off} = 0.25 \times T_c$, where T_c is the expected duration of one locomotion cycle. The delays used for all of our examples fall in the range of 0.07-0.15s. Because of the delays and the internal interconnectivity, the SAN has dynamic properties of its own that are important for generating useful control networks. We view the motions that the SANs produce as being the result of two interacting dynamical systems, that of the SAN and that of the creature itself. One can also think of SANs as being similar in function to the oscillators hypothesized to exist in some animals[13]. When the node delays were removed and the node functions were made more linear in several experiments, it most frequently resulted in a creature that rested in an immobile state after some initial motion. It is perhaps no coincidence that these are exactly the features that make the SANs operate as interesting oscillators by themselves.

5.0 Network Synthesis

We have now defined the construction of creatures and the structure of SANs, but we still need a method to obtain the weight values in the SAN. These will ultimately determine a creature's behaviour. Before outlining our solution, let us briefly consider another possible approach. Consider a top-down method whereby we assume we know the type of motion or desired gait and must now determine the details to generate it. Such an approach has several problems. First, it is often difficult to come up with the desired gait, especially if the creature has no counterpart in nature. Many of our creatures are capable of modes of locomotion that we never could have conceived of beforehand. Second, motion in the desired way might be physically impossible for the creature. We wish to restrict our search to strictly those solutions that are physically feasible. Lastly, the non-linear nature of the creature and its sensing devices makes any such top-down synthesis a very difficult task.

The top-down approach tries to determine a controller given a desired behaviour. The bottom-up approach that we shall take repeatedly generates and evaluates controllers (through simulations) until one or several are found that have desirable behaviours. We thereby exploit the fact that the simulation function is much less expensive than the top-down synthesis function. We have found that the SAN architecture defines a controller space that is highly populated with useful controllers.

Our network synthesis procedure consists of two phases. Phase 1 involves random SAN generation and evaluation. Phase 2 takes the best SAN controllers found in phase 1 and improves upon them. These two phases together define a coarse-to-fine strategy in searching for suitable controllers.

5.1 The Evaluation Metric

An evaluation metric is required for being able to determine the quality of a motion generated by any given SAN. For most of our creatures, this is simply the distance travelled in a fixed amount of time: $f_{eval} = |x(t_{final})|$. The creatures that travel the furthest are usually the ones that have an interesting mode of locomotion. We do usually not care whether the creature moves to the right or the left because both can yield interesting motions. The evaluation metric can be further qualified to be the distance travelled without falling over. For such events we set the evaluation metric to zero. To obtain a controller with energetic hops, the average height of the creature can be incorporated into the evaluation metric: $f_{eval} = \int_0^{t_{final}} (\dot{x} + ky^2)\, dt$. For creatures that should make use of their eye sensors to track a follow-point, the evaluation metric is: $f_{eval} = \int_0^{t_{final}} (v \bullet F)\, dt$, where v is the velocity of the creature and F the unit vector pointing towards the follow point. This metric is a simple measure of how well pursuit is maintained.

5.2 Phase 1: Random Generation and Evaluation

A controller is generated at random by choosing all of its weights at random. This is roughly equivalent to randomly selecting the reactions to be associated with each sensor. A randomly generated

SAN is then evaluated by simulating the behaviour of the creature with the SAN controller and calculating the evaluation metric.

The random search performed by phase 1 is used to discover different possible modes of locomotion for a creature. For this process to be an effective synthesis method, however, we must have some reasonable expectation of finding desirable controllers. At first glance, this does not seem a likely proposition. The space being randomly sampled is very large. For our bounder example it contains 5^{240} possible samples: each of the 240 weights can take any of 5 values. Furthermore, there is no guarantee that any point in this space corresponds to a useful controller.

It is our hypothesis that within the large search space there are many pockets containing useful weight combinations. While the SAN architecture clearly imposes a structure on the solutions, it is not obvious how to search this space. We have found that our dart-throwing strategy is successful a small but significant proportion of the time. The distribution of controllers ranked according to the evaluation metric is a sharply decreasing function with a long tail. Typically 1-5% of the random SANs result in useful motions. This has limitations; as creatures get more complex (6 or more links), fewer candidate motions are aesthetically pleasing.

5.3 Phase 2: Fine Tuning

A second fine-tuning phase may be applied to improve the controllers obtained as a result of the first phase. The fine-tuning phase makes small adjustments to some of the parameters of the SAN and the creature to improve its performance. The weights chosen in the previous step are not among the parameters to be adjusted, however. Because of the non-linear operation of the nodes in the SAN, small changes in the weights often result in either no change at all or a very large change in the dynamics of the system.

The parameters we shall adjust in the fine-tuning phase are those associated with the sensors and the actuators, as well as the delays for the hidden nodes. This list of parameters is shown in Figure 8. We wish to retain the same fundamental mode of locomotion but *evolve* the values of some of the parameters to obtain an improved gait. There are 4 parameters to adjust for each actuator, 2 for each network node, and 2 for each sensor. For our bounder, this results in a total of 44 adjustable parameters.

k1	delay in SAN node for turning on
k2	delay in SAN node for turning off
Amin	minimum desired angle or length for actuator
Amax	maximum desired angle or length for actuator
ks	spring constant for actuator
kd	damper constant for actuator
Smin	lower bound of sensing range for sensor
Smax	upper bound of sensing range for sensor

FIGURE 8. Adjustable parameters for SAN fine-tuning

The approach taken in phase 2 is to make small positive or negative perturbations to randomly chosen parameters. Evaluations are then performed to see if a given change improves the result. We consider two different approaches for performing and evaluating the parameter changes. These are stochastic gradient ascent (SGA) and simulated annealing (SA). These two approaches are distinguished in that one searches for a local optimum, while the second searches for a global optimum.

The structure of the SGA algorithm is shown in Figure 9. The simulated annealing algorithm used is equivalent to that described in [7] and [10]. The simulated annealing algorithm accepts some parameter changes that result in worse performance in order to be able to escape local minima (or maxima in our case). Both algorithms begin with the nominal parameter values assigned when the sensors and actuators were first designed.

```
for (1000 trials)
        randomly choose a parameter to vary
        perturb the parameter value by +delta or -delta
        evaluate the new creature by simulation
        if (creature improved) then
                keep change
        else
                reject change
```

FIGURE 9. Pseudocode for stochastic gradient ascent

A simulated annealing run of 1000 trials was usually sufficient to produce good results. The annealing schedule consisted of 65 evaluations at each annealing temperature, with the next ‘temperature’ being 0.75 times the previous one. In some cases SGA produced better results than SA for the same number of evaluations. For other cases the reverse was true. We expect that simulated annealing would find better solutions given many evaluations and a suitable annealing schedule, while stochastic gradient ascent quickly finds a reasonable local maximum.

6.0 Results

SANs appear to be capable of serving as useful controllers for almost any relatively simple creature. We have successfully experimented with a total of 10 creatures. These creatures are listed in Table 1, and some of their mechanical configurations are given in the Appendix. The creatures are drawn in the vertical plane and make use of the ground to propel themselves forward, with the exception of the fish. The fish makes use of the reaction forces of water to propel itself forward. The crawler consists of four point masses placed in a rectangular configuration. These masses experience less friction sliding forward than backwards. The crawler uses this property to be able to move forward.

Most of the creature designs are the original and only attempts at designing the creature. There remains, however, a certain measure of intuition involved in determining what sensors and actuators might be useful to the creature. The angle ranges of the sensors and actuators for both the bounder and luxo were varied in several experiments before settling on the given choices, which seem to be capable of yielding a broad range of interesting gaits. The forces and torques produced by the SAN controllers are usually not smooth because of the presence of impacts and the use of binary sensors.

TABLE 1. The experimental creatures

creature	links	sensors	actuators	hidden nodes	speed cm/sec
crawler	4	8	2	10	11
fish	4	6	2	5	19
bounder	5	8	4	8	115
luxo	3	6	2	6	79
cart	2	6	1	5	23
walker	6	11	5	6	101
twolink	2	6	1	5	12
threelink	3	8	2	7	33
fourlink	4	11	3	8	55
star	3	8	2	7	9

An evaluation of 200 random controllers (phase 1) finds useful controllers for each of the creatures, although a wider variety of gaits can be obtained by evaluating more. We retain the top 10 results automatically for examination by the user. For most creatures we use the distance moved in 6 seconds as an evaluation metric. A 30 second evaluation is used for the crawler and the fish in order to be able to test their tracking ability. An evaluation of 200 controllers thus requires 1200 seconds of simulation for most creatures. This can take from 1 hour for the cart creature to 6 hours for the walker creature (on a Sun SPARC IPC).

The crawler and the fish were designed to perform tracking. Their purpose in life is to always swim or crawl towards the follow point. They perceive this follow point only through their binary eye sensors. The joint connecting link L4 to L3 in the fish has a passive angular spring and damper. This joint and link L4 serve as the caudal fin of the fish, a necessary feature for efficient swimming. The crawler and the fish are creatures that form a dynamical system with their SANs that is capable of tracking an object successfully. They can do so with binary eyes and have never been given any information on how to move forward, turn, or associate eye information with turning. Figure 10 shows an example of the pursuit motion of the fish.

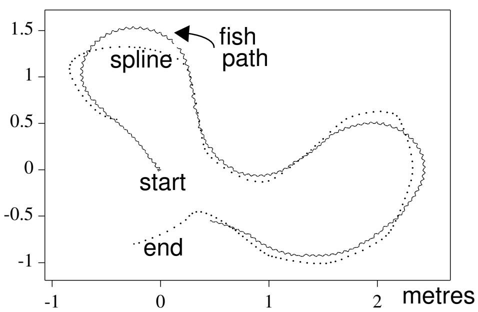


FIGURE 10. The fish chasing a point being dragged along a spline curve.

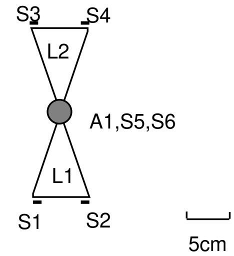
6.1 Variety of Solutions

The most novel modes of locomotion were discovered for very simple creatures, such as those having only two or three links. The cart creature, shown in Figure 11, is perhaps the best example. The creature derives its name from ‘cartwheel’ because it was originally designed with the hope it could move by performing cartwheels. Performing cartwheels is indeed one of the physically-feasible modes of locomotion discovered. The creature is also capable of four other unique and valid gaits, however. All 5 modes are shown in Figure 12. These modes were found in a run of 1000 controller evaluations. Several modes looked promising, but were unable to fully sustain periodic locomotion initially. These motions were easily fixed to yield proper periodic locomotion using the fine-tuning phase as described in the previous section.

A simple two-link chain is also capable of remarkable motions. It is important to note that one of the links is heavier than the other, which is necessary to avoid some situations where the chain is unable to locomote because of its symmetry. One SAN moves the creature forward using a flapping motion that lifts the joint vertex upwards and forwards in small hops. Another performs repeated ‘jumps’ onto its back. Yet another manages to get the links into a position such that it can do a big aerial jump. It ‘falls’ upon landing, but manages to get up to perform another jump, forming a repetitive motion.

More complex figures such as the bounder and the walker produce gaits more familiar to us. The bounder has gaits moving in small hops, big hops, shuffling, and others which are difficult to describe succinctly. The walker moves by performing shuffles, hops, and taking alternating steps. For many objects, the gaits

`mechanical configuration:`



link	mass (kg)
L1	0.10
L2	0.10

`sensors:`

sensor	type	link	min	max
S1	touch	L1		
S2	touch	L1		
S3	touch	L2		
S4	touch	L2		
S5	angle	L2	–180	–120
S6	angle	L2	120	180

`actuators:`

act.	min	max	ks	kd
A1	–140	140	0.002	0.0001

FIGURE 11. The cart creature

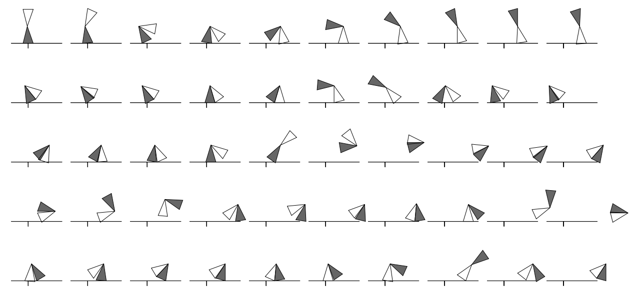


FIGURE 12. Modes of locomotion for the cart

produced are dependent upon the coefficient of friction. This is particularly the case for the three-link and four-link chain creatures.

Many of the solutions not ranking among the best in terms of the evaluation metric can also be interesting. Some creatures fall over but can still succeed in moving forward while on their back. Others move a bit and then become fixed, apart from a repeated twitching motion of leg. This is also the case for creatures that fall over into a state from which they cannot get up. These often repetitively twitch their legs in a style reminiscent of an upside-down insect.

There are limits to the complexity of motion that can be expected to emerge by evaluating randomly-generated controllers. It would be futile, for example, to expect to use our method directly to find a SAN to control a reasonable model of a human body in performing a high-jump. As we will discuss shortly, however, we believe that more complex controllers might be achievable through an evolutionary approach.

6.2 Robustness of Solutions

SANs are also an interesting control structure because they can provide robust control. Figure 13 shows a bounder moving over rough terrain using a SAN that was chosen for its performance over flat terrain. A measure of robustness could be included in the search procedure of phase 1 by performing all the evaluations over terrain of the desired roughness. We believe the robustness is a property related to the inherent simplicity of SAN controllers.

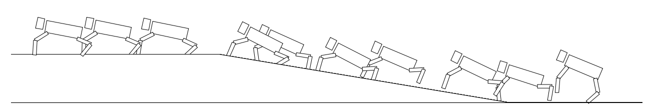


FIGURE 13. The bounder creature climbing a hill

The robustness has limits because the creatures as presently constructed have no means of detecting the upcoming terrain. As such, they are functioning in a manner equivalent to a person walking over rough terrain in the dark. When moving over rough terrain, the same mode of locomotion is usually maintained, with the variations in terrain causing the timing of events to be changed slightly. The gaits are not as robust once they have been fine-tuned. The situation is analogous to running blindly as opposed to walking blindly. Motions optimized for speed are usually less stable.

6.3 Evolution

The second phase of the network synthesis involves making small changes to some parameters of the sensors, actuators, and delays in the SAN. One could also consider including small changes to the actual physical parameters of the system in order to improve its motion. These parameters could include the link lengths, masses, and points of attachment. Such changes are akin to an artificial kind of evolution. For adherance to the constraints of biological systems, the values of these parameters should be interrelated. For example, a stronger spring constant in the actuator, corresponding to a stronger 'muscle', should increase the mass of the appropriate link. Similarly, a larger link should also have an increased mass.

The evolutionary principle could also perhaps be applied to the synthesis of controllers for more complex systems. It is unlikely that random search will stumble upon the best mode of locomotion for a complex articulated figure, such as a good model of a horse. We feel that this is not a large shortcoming of our method because nature herself does not directly arrive at suitable controllers for such creatures. The skeleton, musculature, and control for a horse are the result of a long series of evolutionary changes. We believe it might be possible to arrive at complex controllers by first beginning with the control of simpler figures, such as those that have fewer joints or that have more stable locomotion.

7.0 Conclusions

We have presented a new method that automatically searches for interesting and physically-feasible modes of locomotion for arbitrarily-designed creatures. The solutions are in the form of simple controllers that use binary sensors as input and produce actuator forces and torques as output. Sensor-actuator networks are a compact representation of the complex and varied gaits that they produce. A typical controller has 240 weights that can take on 5 values, and can thus be represented in 70 bytes.

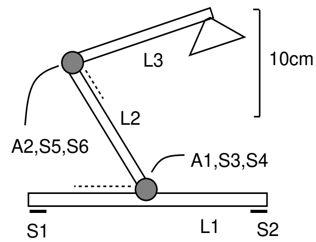
There are several advantages to using SANs. Using the method requires no knowledge of the underlying equations of motion. The user can rapidly construct a new creature and have the method 'discover' several ways it can locomote. It is easy to generate controllers that take advantage of more complex physical phenomena. As an example, the coefficient of friction can have a profound impact on the choice of the best gait. SANs can control mechanical systems that do not always have smooth motion. The user can influence the motion through the specification of the evaluation metric and evaluation terrain in order to achieve the desired speed, energy, height, and robustness of the control. The ability to fine-tune the sensors, actuators, and delays in the SAN means the creature designer need not choose the optimal design parameters to begin with.

SANs have some disadvantages from the point of view of an animator. They do not at present provide the user with as much control over the resulting motion as keyframing or physics-as-constraints methods. Furthermore, there is no guarantee that any useful solution will be found. SANs in their current incarnation do not work very well for systems dominated by linear or smooth dynamics. The synthesis method also fails to find the best modes of locomotion for complex creatures. Our method always results in physically-realizable motions, something that can be an advantage or a restriction in animation.

We believe there are many possible future directions for this work. Many experiments can be carried out using other types of architectures, sensors, and actuators. The fine-tuning phase could be used to determine how to insert passive elastic elements in the system to reduce the amount of expended energy for a given mode of locomotion. The use of SANs as the bottom level of a control hierarchy needs to be investigated. More understanding and analysis of the dynamical systems formed by the SANs and the creatures is needed.

Appendix

mechanical configuration:



sensors:

sensor	type	link	min	max
S1	touch	L1		
S2	touch	L1		
S3	angle	L2	–180	–70
S4	angle	L2	–53	180
S5	angle	L3	–100	75
S6	angle	L3	140	180

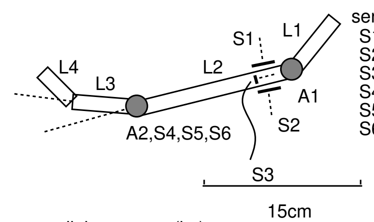
actuators:

act.	min	max	ks	kd
A1	–70	–50	0.05	0.001
A2	60	150	0.04	0.001

link	mass (kg)
L1	0.05
L2	0.10
L3	0.30

Luxo

mechanical configuration:



sensors:

sensor	type	link	min	max
S1	eye	L2	10	180
S2	eye	L2	–10	10
S3	eye	L2	–180	–10
S4	angle	L3	–180	–25
S5	angle	L3	–25	25
S6	angle	L3	25	180

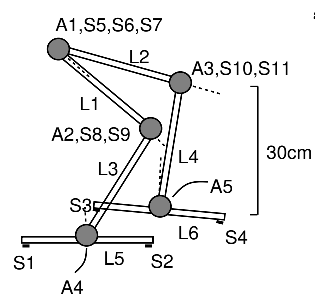
link	mass (kg)
L1	0.2
L2	1.0
L3	0.3
L4	0.1

actuators:

act.	min	max	ks	kd
A1	–20	20	0.003	0.001
A2	–40	40	0.006	0.001

The fish

mechanical configuration:



sensors:

sensor	type	link	min	max
S1	touch	L5		
S2	touch	L5		
S3	touch	L6		
S4	touch	L6		
S5	angle	L2	–180	–15
S6	angle	L2	–15	–5
S7	angle	L2	15	180
S8	angle	L3	–180	–85
S9	angle	L3	–75	180
S10	angle	L4	–180	–85
S11	angle	L4	–75	180

actuators:

act.	min	max	ks	kd
A1	–30	30	0.15	0.01
A2	–105	–70	0.15	0.01
A3	–105	–70	0.15	0.01
A4	20	30	0.10	0.00
A5	20	30	0.10	0.00

link	mass (kg)
L1	0.3
L2	0.3
L3	0.2
L4	0.2
L5	0.1
L6	0.1

The walker

References

[1] N. Badler, B. Barsky, and D. Zeltzer (Eds). *Making Them Move*. Morgan Kaufmann, 1991.

[2] R. Beer. *Intelligence as Adaptive Behavior.* Academic Press, 1990.

[3] V. Braitenberg. *Vehicles: experiments in synthetic psychology.* MIT Press, 1984.

[4] R. A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, 2, 1 (March 1986), 14-23.

[5] L. S. Brotman and A. N. Netravali. Motion Interpolation by Optimal Control. Proceedings of SIGGRAPH ‘88. In *ACM Computer Graphics*, 22, 4 (August 1988), 309-315.

[6] M. F. Cohen. Interactive Spacetime Control for Animation. Proceedings of SIGGRAPH ‘92. In *ACM Computer Graphics*, 26, 2 (July 1992), 293-302.

[7] E. Fiume and M. Ouellette. On distributed, probabilistic algorithms for computer graphics. *Proceedings of Graphics Interface ‘89*, 211-218, 1989.

[8] M. Girard. Constrained Optimization of Articulated Animal Movement in Computer Animation. In *Making Them Move*, Morgan Kaufmann, 1991, 209-232.

[9] J. K. Hodgins, P. K. Sweeney, and D. G. Lawrence. Generating Natural-looking Motion for Computer Animation. *Proc. of Graphics Interface ‘92*, 265-272.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220,13 (May 1983), 671-680.

[11] P. Maes, R. Brooks. Learning to Coordinate Behaviors. *Proc. of AAAI ‘90*, 1990, 796-802.

[12] T. McGeer. Passive Walking with Knees. *Proceedings of the IEEE International Conference on Robotics and Automoation*, 1640-1645, 1990.

[13] M. McKenna and D. Zeltzer. Dynamic Simulation of Autonomous Legged Locomotion. Proceedings of SIGGRAPH ‘90. In *ACM Computer Graphics*, 24, 4 (August 1990), 29-38.

[14] G. S. P. Miller. The Motion Dynamics of Snakes and Worms. Proceedings of SIGGRAPH ‘88. In *ACM Computer Graphics*, 22, 4 (August 1988), 169-178.

[15] W.-Y. Ng. Perspectives on Search-Based Computer-Aided Control System Design. *IEEE Control Systems Magazine*, 13, 2 (April 1993), 65-72.

[16] J. T. Ngo and J. Marks. Spacetime Constraints Revisited. Proceedings of SIGGRAPH ‘93. In *ACM Computer Graphics*, 27 (August 1993).

[17] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine. An Optimal Control Model for Maximum-Height Human Jumping. *J. Biomechanics*, 23, 12, 1185-1198, 1990.

[18] M. H. Raibert and J. K. Hodgins. Animation of Dynamic Legged Locomotion. Proceedings of SIGGRAPH ‘91. In *ACM Computer Graphics*, 25, 4 (July 1991), 349-358.

[19] K. Sims. Artificial Evolution for Computer Graphics. Proceedings of SIGGRAPH ‘91. In *ACM Computer Graphics*, 25, 4 (July 1991), 319-328.

[20] A. J. Stewart and J. F. Cremer. Beyond Keyframing: An Algorithmic Approach to Animation. *Proceedings of Graphics Interface ‘92*, 273-281, 1992.

[21] M. van de Panne, E. Fiume, and Z. G. Vranesic. A Controller for the Dynamic Walk of a Biped Across Variable Terrain. *Proceedings of the 31st IEEE Conference on Decision and Control*, 1992.

[22] M. van de Panne, E. Fiume, and Z. G. Vranesic. Reusable Motion Synthesis Using State-Space Controller. Proceedings of SIGGRAPH ‘90. In *ACM Computer Graphics*, 24, 4 (August 1990), 225-234.

[23] J. Wilhelms and R. Skinner. An Interactive Approach to Behavioral Control. *Proceedings of Graphics Interface ‘89*, 1-8, 1989.

[24] A. Witkin and M. Kass. Spacetime Constraints. Proceedings of SIGGRAPH ‘88. In *ACM Computer Graphics*, 22, 4 (August 1988), 159-168.

FIGURE 14. Bounders on the run

FIGURE 15. Walking on campus