MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Towards Practical Automated Motion Synthesis

J. Auslander, A. Fukunaga, H. Partovi, J. Christensen, L. Hsu, P. Reiss, A. Shuman, J. Marks, J.T. Ngo

TR94-11 December 1994

Abstract

We extend an earlier motion-synthesis algorithm for physically realistic articulated figures in several ways. First, we summarize several incremental improvements to the original algorithm that improve its efficiency significantly and provide the user with some ability to influence what motions are generated. These techniques can be used by an animator to achieve a desired movement style, or they can be used to guarantee variety in the motions synthesized over several runs of the algorithm. Second, we report on new mechanisms that support the concatenation of existing, automatically generated motion controllers to produce complex, composite movement. Finally, we describe initial work on generalizing the techniques from 2D to 3D articulated figures. Taken together, these results illustrate the promise and challenges afforded by the automated motion-synthesis approach to computer animation. Key Words: Animation, spacetime constraints, heuristic methods, machine learning, stochastic optimization, evolutionary computation.

ACM Transactios on Graphics

1. First printing, TR-94-11, June 1994.

1 Introduction

Automatic motion synthesis for articulated figures is the problem posed by the Spacetime Constraints (SC) paradigm for animation [19]. In this paradigm, the animator specifies only the physical structure of an articulated figure and quantitative criteria for success in a desired task. The computer must compute physically realistic motion for the figure that is near optimal[1] according to the task criteria.

Early motion-synthesis algorithms used local optimization to refine initial figure trajectories by making them more compliant with physical law, or by improving the motion with respect to the task criteria [4, 5, 19]. However, local optimization has inherent limitations for this problem: it is usually confounded by the discontinuities and local optima found in the search space of a typical SC problem, and it leaves primary responsibility for constructing coarse initial trajectories with the human animator.

Recently, a new approach to the motion-synthesis problem has been proposed. The approach is not to compute the figure's trajectory directly, but instead to generate automatically a *motion controller* that, when executed, will produce the desired motion [10, 11, 15, 16]. In any particular embodiment of this approach, two broad and nearly independent choices must be made:

1. how the motion controller is to be represented; and
2. how the space of possible controllers is to be searched.

An early example of this idea from the computer-graphics literature is provided by Ridsdale [15], who used a neural network to control a simulated one-rod robot arm that could hit a ball against the wall of a handball court. Ridsdale's search strategy is an adaptation of the standard back-propagation algorithm for training neural networks, augmented with simulated annealing to avoid local minima.

Van de Panne and Fiume [16] have described a similar approach that is capable of solving more general motion-synthesis problems. In their work a motion controller is termed a sensor-actuator network (SAN). In a SAN, the actuators' responses are interdependent nonlinear functions of the figure's physical-state variables. Van de Panne and Fiume search the space of possible controllers in two stages. The first stage is a random generate-and-test procedure, and the second stage effects a subsequent refinement by simulated annealing or stochastic gradient ascent.

In contrast to these explicitly connectionist approaches, Ngo and Marks employ a bank of mutually independent controllers called stimulus-response (SR) rules. Information from the physical environment is used principally to determine which rule is active at any given time in the physical simulation. We refer to this as a banked stimulus-response (BSR) controller. The space of possible BSR controllers is searched by evolutionary computation [7, 10, 11].

There are several reasons why the automatic synthesis of motion controllers is not yet a practical approach to motion synthesis. In this paper we focus on four of these limitations in the context of the previously reported BSR-controller work:

1. The original search algorithm is slow and complex—it can require up to an hour on a massively parallel computer to compute a motion controller.
2. There is no mechanism to influence the search algorithm to produce motions that match an animator's preconceptions—the algorithm is inherently random, and, in general there is no way to affect or even predict what it will produce in any given run.
3. The motion generated by a single controller is relatively simple—the only way to get complex, composite motion is to concatenate several problem instances in time and to generate separate motion controllers serially for each subproblem. This time-consuming approach is similar to one proposed by Cohen [5] in the context of the original Spacetime Constraints paradigm [19].
4. The existing algorithm works for 2D articulated figures only—generalizing to 3D would appear to be computationally formidable.

[1] Because this optimization problem is NP-hard, there exists no polynomial-time algorithm that is guaranteed to return optimal solutions unless P=NP.

We show how each of these problems can be addressed to some degree. We begin by summarizing several incremental improvements to the original algorithm that obviate the need for parallel computation, improve efficiency significantly, and afford the user limited control over the search process. An animator can use this latter capability to influence directly the style of the resulting motions; it can also be used to produce a suite of qualitatively different motion controllers for a given SC problem. A selection of different motion controllers can be passed as input to an editing module in which the animator can concatenate controllers interactively to produce composite motions. We discuss and demonstrate two different ideas for supporting interactive controller concatenation. We conclude by reporting initial results on automatic motion synthesis for 3D articulated figures.

2 The Original Approach and Extensions

2.1 Efficient automatic synthesis of BSR motion controllers

The system described originally by Ngo and Marks [10, 11] searched in a space of BSR controllers using a massively parallel genetic algorithm (GA) that ran on a 4096-processor CM-2. In more recent work, Fukunaga et al. [7] have shown that a serial search algorithm, also based on the principles of evolutionary computation, exhibits far better performance. Although we assume general familiarity with the approach as described in previous work, we now briefly review the details of the BSR controller and the serial search algorithm.

A BSR controller governs a vector $\vec{\theta}(t)$ of joint angles, given information about the physical environment in the form of a vector $\vec{S}(t)$ of *sense variables*. Sample sense variables for an articulated figure are listed in Table 1.

$\theta_1, \theta_2, \ldots, \theta_{n-1}$	Joint angles
$f_1, f_2, \ldots, f_{n+1}$	Contact forces at rod endpoints
y_{cm}	Height of center of mass
$\dot{y}_{\mathrm{cm}}$	Vertical velocity of center of mass

Table 1: Components of the vector $\vec{S}$ of sense variables for an n-rod articulated figure.

The controller contains R stimulus-response rules. Each rule i is specified by stimulus parameters $\vec{S}^{\mathrm{lo}}[i]$ and $\vec{S}^{\mathrm{hi}}[i]$, and response parameters $\vec{\theta}^0[i]$ and $\tau[i]$. Based on the instantaneous value of the sense vector $\vec{S}(t)$, exactly one rule is active at any one time. In particular, each rule i receives a score based on how far the instantaneous sense vector $\vec{S}(t)$ falls within the hyperrectangle whose corners are $\vec{S}^{\mathrm{lo}}[i]$ and $\vec{S}^{\mathrm{hi}}[i]$. The highest-scoring rule is said to be marked *active*. (If $\vec{S}(t)$ is not inside the hyperrectangle associated with any rule, the rule active in the previous time step remains active.) The joint angles $\vec{\theta}(t)$ are made to approach the target values $\vec{\theta}^0[i_{\mathrm{active}}]$ prescribed by the active rule i_{active}.

The following pseudocode summarizes how a BSR controller behaves and is evaluated:

Set i_{active} to 1
for $t = 1$ **to** $t_{\max}$
  Cause joint angles $\vec{\theta}(t)$ to approach $\vec{\theta}^0[i_{\mathrm{active}}]$ with time constant $\tau[i_{\mathrm{active}}]$
  Simulate motion for time interval t
  Measure sense variables $\vec{S}(t)$
  Possibly change i_{active}, based on $\vec{S}(t)$
end for
Assign the controller a *fitness value* based on how well
  the simulated motion meets the animator-supplied task criteria

The search algorithm in Figure 1 is used to compute effective BSR controllers. This algorithm works by simulating the application of several hill climbers in parallel to a working set of motion controllers.

```
Initialize a set of random motion controllers
Evaluate each motion controller in the set
for i = 1 to number_of_iterations
  for each individual controller in the set do
    Mutate (i.e., reinitialize or perturb) the controller
    Evaluate the new controller
    if the new controller is better than the old one then
      Replace the old controller with the new one
  end for
  if (i mod reseed_interval) = 0 then
    Rank order the set of controllers
    Replace bottom 50% of the set with top 50%
  end if
end for

Reasonable parameter values:
  set_size = 10
  reseed_interval = 200
  number_of_iterations = 40,000
```

Figure 1: Serial search algorithm.

Periodically, the set is "reseeded" by replacing the worse half of the set with copies of the better half. This refocuses the search on more promising areas of the search space. The running time for this algorithm is typically 3–6 minutes on a single DEC 3000/400 AXP workstation for 2D articulated-figure controllers like those illustrated in Figures 5–10.[2]

The most important factor in the success of the search algorithm is the fact that it is searching in the space of possible motion controllers, rather than the space of trajectories. Nevertheless, certain secondary details of the initial random-generation process and the mutation operator are important for successful motion synthesis and are described in detail elsewhere [10, 11].

2.2 Additional fitness terms

The inability of the user to influence the search algorithm in a principled and organized fashion is a potential shortcoming of fully automated approaches to motion synthesis. This causes difficulties when the animator has a preconceived idea for how an animated character should move, but is unable to cause the search algorithm to generate the expected motion. It is also a problem when what is required is not just a controller for one kind of motion, but a suite of controllers for many different kinds of motion for the same animated character (§3).

A concrete instance of this problem is provided by Mr. Star-Man. Given the task of making Mr. Star-Man travel, the search algorithms have produced a variety of motions, including a shuffling gait (Figure 5) and a cartwheel (Figure 6). However, since the shuffling gait tends to cover more distance per unit time than the other motions, it is the motion most frequently produced. As a test case, we set out to provide some mechanism whereby the search could reliably be guided toward either the shuffle or the cartwheel.

For any optimization task, an obvious way to obtain alternative solutions is to change the fitness function arbitrarily, assuming that one's optimization technique can cope with arbitrary fitness functions. Early experimentation showed that this approach is feasible within the BSR paradigm. However, as a general

[2] Ngo and Marks originally reported running times of 30–60 minutes on a CM-2 Connection Machine to compute comparable BSR motion controllers [10, 11]. Van de Panne and Fiume reported running times of 1–6 hours on a Sun SPARC IPC for their algorithm [16].

Term	*Function*
`max_cm_height`	Maximum height of center of mass during motion
`slip_sum`	Distance traversed by body parts in continuous contact with ground
`rotations`	Number of full-body rotations during motion

Table 2: Secondary terms in the fitness function.

technique it is too arbitrary: the animator should be given some guidance as to how the fitness function can be modified most effectively.

Our attempt at a more structured approach to fitness-function modification involves combining a single primary term with one or more of a suite of secondary terms. The *primary term* in the fitness function is the one given the most weight, and is therefore the one that determines the most salient characteristics of the motion. For example, to get Mr. Star-Man to travel, *i.e.*, to move from left to right, the primary term in the fitness function is the horizontal distance traversed by his center of mass. Primary terms are determined by the requirements of the animation script, and are of necessity quite arbitrary. The *secondary terms*, which are added to the primary term in the fitness function, determine minor characteristics of the motion. Our goal was to describe a canonical set of secondary terms that would be broadly useful, regardless of the primary term. Our initial candidate set of secondary terms for 2D articulated figures is given in Table 2.

For the sample problem we proposed above, assigning a positive weight only to the secondary term `slip_sum` guarantees a shuffling gait, whereas assigning a positive weight only to the secondary term `rotations` guarantees a cartwheel. For our other animated characters, varying the coefficients of the small set of terms in Table 2 was sufficient to generate reliably all motions that we had ever seen occur at random, or that we had anticipated from a priori considerations. Moreover, the motions generated correlate qualitatively with the secondary fitness terms: a positive coefficient for `max_cm_height` biases the search in favor of hopping or jumping motions, a positive coefficient for `slip_sum` encourages sliding or shuffling, and a positive coefficient for `rotations` usually leads to some kind of gyration.

2.3 Different sense variables

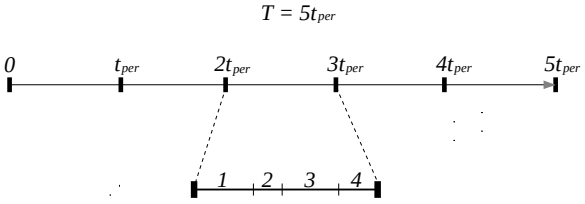
The nature of the motions produced by controller synthesis depends intimately on the space of controllers made available for searching. For example, BSR controllers produce motions that are quite different from those produced by SANs, which tend to exhibit characteristic sinusoidal vibrations [12, 17]. Thus, another way to influence the style of generated motions might be to modify the form of the BSR controller.

In a *time-based* BSR motion controller, the sole input "sensor" measures elapsed time, so that the bank of stimulus-response rules is equivalent to a simple script of responses that is performed one or more times. (We refer to the original BSR controllers described in §2.1 as *sense-based* to distinguish them from this new kind of motion controller.) In our form of the time-based BSR controller (Figure 2), a time interval with user-defined length t_{per} is broken into a small number R of mutually exclusive time segments. During each segment, the corresponding stimulus-response rule is active. The sequence of time segments is repeated to fill the length of the simulation, T, which is also chosen by the user. Thus, if $t_{\text{per}} < T$, then the sequence of actions specified by the rules is repeated periodically. If, on the other hand, $t_{\text{per}} \geq T$, then the action sequence is not constrained to be periodic.

In recent papers [10, 11, 16] it has been argued or assumed that the space of controllers with access to information about physical state should be easier to search than the space of controllers based on time alone. Unexpectedly, the space of time-based motion controllers for 2D articulated figures[3] proved easier to search than the space of sense-based controllers.[4] Furthermore, by manually varying the value of the period t_{per},

[3] The issue is more complex for 3D articulated figures, as we describe in §4.

[4] Good time-based controllers for these problems can be found in a few tens of seconds on a DEC 3000/400 AXP workstation



Rule	*Stimulus*	*Response*
1	$[0, t_1^{\text{end}})$	$(\vec{\theta}_1^0, \tau_1)$
2	$[t_2^{\text{begin}} = t_1^{\text{end}}, t_2^{\text{end}})$	$(\vec{\theta}_2^0, \tau_2)$
3	$[t_3^{\text{begin}} = t_2^{\text{end}}, t_3^{\text{end}})$	$(\vec{\theta}_3^0, \tau_3)$
4	$[t_4^{\text{begin}} = t_3^{\text{end}}, T)$	$(\vec{\theta}_4^0, \tau_4)$

Figure 2: An illustration of a time-based BSR motion controller, with $R = 4$.

it is possible to elicit a variety of motions, from highly periodic to completely aperiodic; thus time-based controllers afford the animator straightforward control over the periodicity of the computed controller. For example, the "tumbling" sequence of Beryl Biped depicted in Figure 7 was computed with $t_{\text{per}} = T$, where T is the length of time available for the entire motion. The periodic shuffling in Figure 8 was computed with a value of $t_{\text{per}} \ll T$.

3 Composite Motion Synthesis

The automatic motion-synthesis techniques described here and elsewhere are currently capable of producing controllers only for relatively simple motions. To address this limitation in the context of the Witkin-Kass approach [19], Cohen [5] implemented a system that permits a human animator to design and refine a trajectory by interactively submitting simple SC subproblems for solution on-line. An analogous system for interactive controller synthesis may be quite attractive for 2D articulated figures, especially given the ability to solve 2D SC problems in times ranging from minutes to seconds on a modern workstation. However, such speedy turnaround is not likely for 3D SC problems in the near future (§4), so other approaches for generating complex, composite motions are needed.

As an alternative to Cohen's approach, we propose the following regimen for creating composite motion sequences for 2D articulated figures:[5]

1. The animator defines the animated character by specifying the physical structure of an articulated figure.

2. The computer generates a suite of different sense-based motion controllers for this character automatically off-line, using the serial search algorithm (Figure 1) and various combinations of primary and secondary fitness terms (§2.2).

3. The animator develops composite motions by interactively concatenating selected controllers in time using an animation editor.

using the serial search algorithm described in Figure 1. Two factors that may contribute to the ease with which good time-based controllers can be found is the relatively small number of parameters per rule, and the relatively small number of rules per controller.

[5] This regimen is very similar in spirit to one described previously by van de Panne et al. [18], though the motion-controller representations in both schemes are very different, and controllers are concatenated automatically in our approach. Indeed, the broad idea of concatenating motion controllers has been proposed and investigated previously in a variety of contexts [2, 3]; our contribution is distinguished by the method we use for concatenating controllers automatically and the scope and generality of the motion controllers considered.

Unlike Cohen's approach, this method has the advantage that all lengthy computations are performed off-line in step 2; the animator's interaction with the editor in step 3 does not require the solution of additional SC problems. However, it is not immediately obvious how the concatenation operation in step 3 can be supported: if a motion controller is invoked for an articulated figure that starts in a different configuration than the one for which the controller was designed, what happens?

One possibility is that this is not a problem, due to the intrinsic robustness in standard sense-based BSR controllers. If a figure's physical configuration (*i.e.*, its joint angles, location, orientation, and linear and angular momentum) is reasonably close to a configuration that occurs somewhere in its normal trajectory (the one followed by the figure when started from its expected initial configuration), then the motion controller will usually produce a motion similar to the one it was designed to produce. Thus the animator need not be intolerably precise in choosing when to switch from one motion controller to another in order to get a desired composite motion.

We tested this hypothesis by building a graphical editor for animation that allows a user to switch between precomputed, sense-based motion controllers interactively, using a direct-manipulation interface. This simple mechanism proved to be usable. If the animator took the time to become familiar with the various motion controllers at his disposal and was willing to explore alternative transition points between controllers patiently and intelligently, then it was usually possible to produce a desired composite motion, involving up to five different controllers, in a few minutes [6].

Nevertheless, this simplistic approach undoubtedly requires more user input and expertise than is desirable, because the motion controllers are not infinitely robust. For example, the typical result of a careless concatenation of motion controllers is shown in Figure 9. Three previously computed controllers—one for cartwheeling, one for jumping, and one for shuffling—have been linked together in series, but the transition from the cartwheel to the jump is flawed: the jumping controller is capable of effecting a jump when Mr. Star-Man is more or less upright, but fails otherwise. When concatenating controllers in the animation editor, the animator usually starts out in such a situation, and then attempts to repair it by testing different transition points between the different controllers.

We have automated aspects of this otherwise tedious transition-point-selection process by implementing three techniques [1]:

- *Enhanced motion controllers:* By randomly perturbing the initial physical configuration of the articulated figure during each iteration of the original motion-synthesis process, the resulting sense-based motion controller can be made more robust. (The use of controllers made more robust by this technique also makes transition-point selection easier to do in the manual animation editor described above.)

- *A composite-motion scripting language:* To determine an optimal set of transition points, there must be some optimization criterion. Using a scripting language, the animator quantifies the desired characteristics of each phase of the composite motion. An interpreter translates these characteristics into a single fitness function suitable for optimization. A sample script is shown in Table 3.

- *A heuristic search strategy:* Beginning with an initial guess at a suitable set of transition points, a greedy search algorithm iteratively moves each transition point in the direction that most improves the fitness function encoded in the composite-motion script, terminating when no further local improvements are possible.

Thus in this approach the animator is responsible for entering an initial set of transition points and a composite-motion script (both of which are entered via a text editor in our current implementation), but is not responsible for adjusting the transition points.

Given the script in Table 3 and the initial concatenation of motion controllers depicted in Figure 9, our technique adjusted the transition points automatically to achieve the desired composite motion (consisting of a cartwheel, a jump,[6] and a shuffle) depicted in Figure 10.

[6] This unusual jump is made possible by applying high torque to bring the legs together quickly. This may not be very plausible biologically; however, it is physically correct, given that we did not limit the amount of torque that could be applied by Mr. Star-Man's thigh muscles.

Script:

```
cartwheel   :hlog(product(delta(var(cm_x)),
                    equal(delta(var(time)),159,10)));
jump        :hlog(product(best(var(min_y)),
                    equal(delta(var(time)),90,10)));
shuffle     :hlog(product(delta(var(cm_x)),
                    equal(delta(var(time)),100,10)));
```

Translation:

- Credit for the cartwheel phase is proportional to the horizontal distance traveled by the center of mass (`cm_x`) times a factor that penalizes deviation from the original user-supplied duration of the cartwheel (159 simulator ticks).
- Credit for the jump is a function of the highest height cleared during this phase (`min_y`) times a factor that penalizes deviation from the original duration of the jump (90 simulator ticks).
- Credit for the shuffle phase is proportional to the horizontal distance traveled by the center of mass (`cm_x`) times a factor that penalizes deviation from the original duration of the shuffle (100 simulator ticks).
- The credit for the three terms is combined according to the function:

$$\text{credit} = \log(1 + \text{credit}_{\text{cartwheel}}) + \log(1 + \text{credit}_{\text{jump}}) + \log(1 + \text{credit}_{\text{shuffle}})$$

Table 3: A sample composite-motion script.

4 Initial Experiments with 3D Articulated Figures

The controller-synthesis approach has produced exciting results, but only for 2D articulated figures. An open question is whether this approach will generalize to 3D. There is every reason to believe that the generalization to 3D should be very difficult. The dimensionality of the controller space approximately doubles for a creature with a given number of joints, primarily because each joint can now have two degrees of freedom. More importantly, for many 3D articulated figures a large fraction of the search space is occupied by controllers that cause the figure to lose balance and fall over unrecoverably. This problem is much less severe in 2D. Despite these caveats, we have been able to generate automatically effective motion controllers of the BSR variety for a selection of 3D motion-synthesis problems. Our initial results are reported below.

4.1 Time-based controllers

Our research in motion synthesis for 3D articulated figures began with an investigation of time-based BSR motion controllers. As indicated in §2.3, this kind of motion controller is simpler to code, makes for an easier search problem, and seems to be just as able to generate useful simple motions for 2D articulated figures as the original sense-based BSR controllers. This also proved true for stable 3D articulated figures, but not for unstable ones (§4.2).

A 3D time-based BSR controller is similar in form to its 2D counterpart (Figure 2), except that a response for rule i is an ordered triple $(\vec{\theta}_i^0, \vec{\psi}_i^0, \tau_i)$, where $\vec{\theta}_i^0$ and $\vec{\psi}_i^0$ are a set of target Euler angles for all the joints in the articulated figure. Thus, a time-based BSR controller contains $R-1$ independent variables to divide up each period of time t_{per}, $R(2n-2)$ target Euler angles (where n is the number of rods in the figure), and R time constants, giving a total of $2nR-1$ variables. (The value t_{per} is also essential to the specification of the controller, but because it is set by the user we do not count it as an independent parameter.) For example, Rex (depicted in Figure 12) has $n = 11$ rods and $R = 4$ SR rules in his motion controller, so the total number of variables is 87. It is the task of the search algorithm to find 87 floating-point values for these variables that will cause the figure to achieve the desired objective, as expressed in the fitness function. (We used a minor variant of the serial search algorithm described in Figure 1 [13, 14].)

Figure 11 shows a trajectory that is typical of those produced by time-based BSR motion controllers. Cujo, a dog-like creature, propels himself forward by bounding repeatedly. A trajectory for Rex, depicted in Figure 12, is the result of separate motion-synthesis problems that were solved seriatim (*i.e.*, using an approach analogous to Cohen's [5]): Rex learns to walk; then given the final state of that walk he learns to turn; then given the final state of the turn he learns to walk again. The fitness function for Cujo was simply the distance traveled by his center of mass. Rex's fitness function was similar, but it also included secondary terms that penalized sideways motion and falling down.

The progress of the search algorithm in finding one of Rex's motion controllers for walking is shown in Figure 3: the top curve depicts the fitness of the best time-based controller found so far, plotted against the number of controller evaluations performed. In fashion typical for the articulated figures considered here, rapid progress is made through the first 50,000 evaluations, after which progress is generally slower. One evaluation of a motion controller for Rex requires 1,000 time steps of simulation, and about 0.3 seconds of elapsed time on a Digital 3000/400 AXP workstation. An acceptable controller had therefore been found in just under five hours.

4.2 Sense-based controllers

The simplicity and power of time-based BSR motion controllers make them very attractive when compared to the original sense-based BSR controllers. Unfortunately, there is some evidence to suggest that the time-based approach may have inherent limitations. The most difficult 3D motion-synthesis problem we have considered is that of making Bob the Biped walk. Bob, a biped with point feet (see Figure 13), is very unstable, so he falls over easily. To date, we have been unable to generate automatically a time-based motion controller for Bob that allows him to walk more than two or three steps before keeling over.[7]

[7] Bob's point feet make the task he faces similar to learning to walk on stilts. With real human feet, the difficulty of walking is considerably diminished by the flexibility of one's ankles in response to contact with the ground. Because a figure's internal deformations are determined kinematically in our simulator, a controller representation that permits "flexible ankles" would

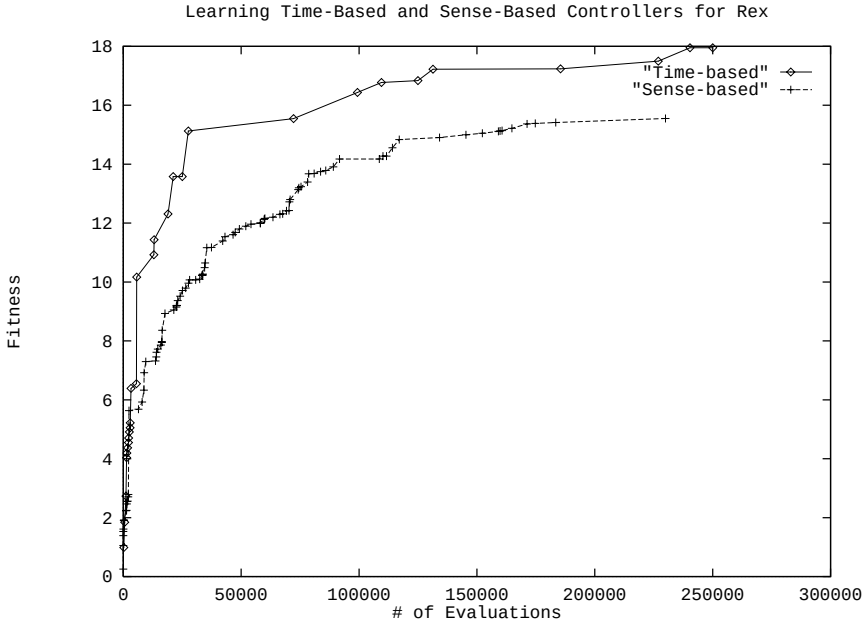


Figure 3: Learning curves for both kinds of motion controller.

However, we have been successful in computing useful sense-based motion controllers for Bob. The physical senses used in our 3D controllers are the height of Bob's center of mass and its linear velocity, Bob's angular velocity, an up vector (the latter three senses are expressed as 3D vectors in a figure-centric coordinate system), contact forces and collision impulses for each possible contact point, and internal Euler angles for each flexible joint. The increased complexity of each SR rule and the desirability of having more rules (typically 10) in a sense-based BSR controller means that 880 floating-point numbers are used to specify a sense-based motion controller for Bob the Biped. The corresponding number for Rex (Figure 12) is 1,280.

These differences in the underlying BSR representation necessitate some minor, but important changes to the search algorithm. The basic form of the algorithm remains the same (Figure 1), but the initialization and mutation procedures require modification. Every rule in a time-based controller is guaranteed to be executed during each time period of duration t_{per}, but no such a priori guarantee can be made for a sense-based controller. In fact, the addition of a randomly generated SR rule to an existing sense-based controller is exponentially unlikely to affect the generated motion, because a randomly generated stimulus region occupies an exponentially small fraction of the full volume of the sense space. Therefore, to ensure that non-creep mutations (*i.e.*, those that generate a new rule from scratch, as opposed to those that make a small change to an existing rule [10, 11]) play a useful role, it is necessary to devise more aggressive *relevance heuristics*, which are methods for generating random stimulus regions that are guaranteed to affect the generated motion. The relevance heuristic employed in the original work on BSR controllers [10, 11] prescribes that a newly generated stimulus hyperrectangle share a "corner" in sense space with the sense-space trajectory generated by the unmutated controller. With 3D figures, it was necessary to change this relevance heuristic to require that the hyperrectangle *contain* some point on the trajectory. Applying three rounds of this mutation operation to the first set of 100 motion controllers was also found to be a useful strategy for enriching the initial population, as was the application of a low-pass filter to the values of all the physical senses to remove noise and high-frequency variation [13, 14].

While sense-based motion controllers generally attained fitness scores inferior to those achieved by time-based controllers for stable articulated figures (and also took longer to find—see Figure 3), they were dis-

be difficult to design. The additional computational cost of a more general physical simulator able to handle these phenomena might place results comparable to those reported here beyond the reach of our current serial hardware.

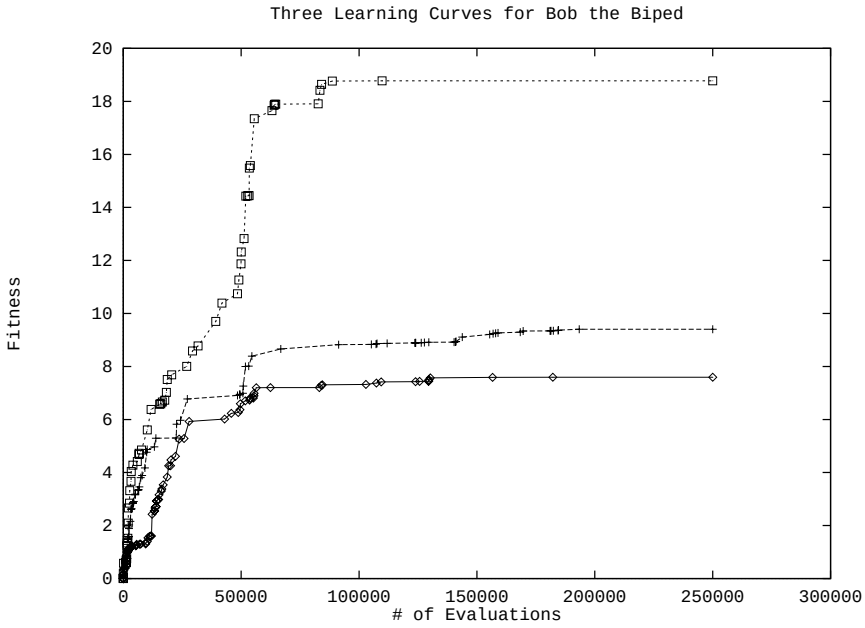


Figure 4: Variability in search-algorithm performance.

tinctly superior for Bob the Biped. The walk depicted in Figure 13 was one of two effective, forward-facing walking strategies discovered.[8] (The strategy not shown here is a highly periodic shuffling motion.) We also noticed a considerable variation in the quality of the BSR controllers found by different runs of the same algorithm when physical sensors were used (see Figure 4), which suggests that multiple runs of the algorithm, or a larger population of candidate solutions, might be best for 3D articulated-figure motion-synthesis problems with this level of difficulty.

5 Conclusions and Further Work

This paper recounts a suite of empirical studies we have undertaken to explore and enhance the practicality of the BSR-controller approach to automatic motion synthesis for 2D and 3D articulated figures. These studies have not led to definitive conclusions, but they have served to indicate and clarify directions for further research in this area. In particular, the research described here raises the following issues:

- *Is there a substantially faster way to compute BSR-style motion controllers?* Even though we managed to reduce the time needed for 2D motion synthesis to acceptable levels (less than 10 minutes for sense-based controllers, and less than two minutes for time-based controllers), 3D motion synthesis for simple tasks can still take hours of workstation time. Although an earlier attempt to use massive parallelism for motion synthesis was not especially effective [9, 10, 11], a renewed investigation of parallel search for motion synthesis may be worthwhile.

- *How useful are time-based BSR controllers?* Time-based BSR controllers are simple and easy to compute. They also afford the animator an easy mechanism for determining the periodicity of the resulting

[8] To our amusement (and then to our annoyance), several controllers were computed that achieved net forward movement, but in such a way that Bob either gyrated or faced backward through much of the motion. To eliminate this behavior and thereby encourage more conventional forward-facing walking, we modified the fitness function to penalize trajectories in which Bob experienced a net rotation about the vertical axis. (In addition to this rotation penalty, the fitness function had already included secondary terms to reward forward motion by both the center of mass and the feet, and to penalize sideways motion and falling over.)

motion. Unfortunately, they have two major drawbacks: they are not easily concatenated to give composite motions, and they appear to be of limited utility for unstable 3D articulated figures. If these latter issues could be addressed, time-based controllers might be preferred to sense-based controllers for animation purposes.

- *Is there a canonical set of useful secondary fitness terms?* Currently, creating animations via automatic motion synthesis involves much modification of fitness functions. If this remains a completely ad hoc process, the appeal of this approach will be limited. We have suggested the use of canonical secondary fitness terms to influence the ancillary characteristics of the generated motions in a systematic way. However, it is clear that if this idea is to be useful, the set of terms in Table 2 must be expanded considerably: witness the various secondary terms we needed to use to influence the motions of the 3D figures discussed in §4.

- *What is the best way to generate composite motions?* The ability to generate composite motions is essential to achieving any kind of practical utility for automatic motion synthesis. We have proposed that motions be composed by interactive concatenation of simple BSR motion controllers. The ideal editor to support this process would be interactive and easy to use, and would provide some automated support for finding near-optimal concatenations. It would be necessary to refine our initial prototypes to get a reasonably effective tool; this may not be easy, especially for 3D motion controllers.

- *Are BSR-style motion controllers suitable for 3D motion synthesis?* Our limited initial experimentation suggests the answer is yes. Furthermore, we have also experimented successfully with 3D BSR motion controllers for mass-spring models [14]. However, the degree of collective experience with automatic motion synthesis is not so great that we can assert with confidence that this kind of controller is superior to all others; certainly for some motion-synthesis tasks, other approaches will be more useful.

Finally, it must be noted that all our work in 2D and 3D has used, for reasons of computational speed, a simplified physical simulator similar to one described by Hahn [8, 10, 13]. As computational resources permit, the issues above should be addressed in the context of more realistic physical simulation.

6 Acknowledgments

The support of the following companies and foundations is acknowledged: Digital Equipment Corporation, the Fannie and John Hertz Foundation, Walter Hewlett and Hewlett-Packard, and the National Science Foundation. We thank Bud Lassiter and Jacob Bricca of Interval Research Corporation for extensive assistance with the accompanying video.

References

[1] J. Auslander, J. Marks, and J. T. Ngo. Composite motion synthesis. In preparation, 1994.

[2] N. I. Badler, B. A. Barsky, and D. Zeltzer, editors. *Making Them Move: Mechanics, Control, and Animation of Articulated Figures.* Morgan Kaufmann, San Mateo, CA, 1991.

[3] N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating Humans: Computer Graphics Animation and Control.* Oxford University Press, Oxford, England, 1993.

[4] L. S. Brotman and A. N. Netravali. Motion interpolation by optimal control. *Computer Graphics*, 22(4):309–315, August 1988.

[5] M. F. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26(2):293–302, July 1992.

[6] A. Fukunaga, L. Hsu, P. Reiss, A. Shuman, J. Christensen, J. Marks, and J. T. Ngo. Motion-synthesis techniques for 2D articulated figures. Technical Report TR-05-94, Center for Research in Computing Technology, Harvard University, March 1994.

[7] A. Fukunaga, J. Marks, and J. T. Ngo. Automatic control of physically realistic animated figures using evolutionary programming. In *Proceedings of the Third Annual Conference on Evolutionary Programming (EP94)*, San Diego, CA, February 1994. In press.

[8] J. K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4):299–308, August 1988.

[9] J. T. Ngo and J. Marks. Massively parallel genetic algorithm for physically correct articulated figure locomotion. Working Notes for the AAAI Spring Symposium on Innovative Applications of Massive Parallelism, Stanford University, March 1993. Available as Technical Report SS-93-04 from AAAI Press.

[10] J. T. Ngo and J. Marks. Physically realistic motion synthesis in animation. *Evolutionary Computation*, 1(3):235–268, 1993.

[11] J. T. Ngo and J. Marks. Spacetime constraints revisited. In *SIGGRAPH '93 Conference Proceedings*, pages 343–350, Anaheim, CA, August 1993. ACM SIGGRAPH.

[12] J. T. Ngo and J. Marks. Spacetime constraints revisited. ACM SIGGRAPH Video Review, Issue 96: Video Supplement to the SIGGRAPH '93 Conference Proceedings, 1993.

[13] H. Partovi. Motion synthesis for 3D articulated figures. Harvard University undergraduate thesis, April 1994.

[14] H. Partovi, J. Christensen, A. Khosrowshahi, J. Marks, and J. T. Ngo. Motion synthesis for 3D articulated figures and mass-spring models. Technical Report TR-06-94, Center for Research in Computing Technology, Harvard University, March 1994.

[15] G. Ridsdale. Connectionist modelling of skill dynamics. *The Journal of Visualization and Computer Animation*, 1:66–72, 1990.

[16] M. van de Panne and E. Fiume. Sensor-actuator networks. In *SIGGRAPH '93 Conference Proceedings*, pages 335–342, Anaheim, CA, August 1993. ACM SIGGRAPH.

[17] M. van de Panne and E. Fiume. Sensor-actuator networks. ACM SIGGRAPH Video Review, Issue 96: Video Supplement to the SIGGRAPH '93 Conference Proceedings, 1993.

[18] M. van de Panne, E. Fiume, and Z. Vranesic. Reusable motion synthesis using state-space controllers. *Computer Graphics*, 24(4):225–234, August 1990.

[19] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988.

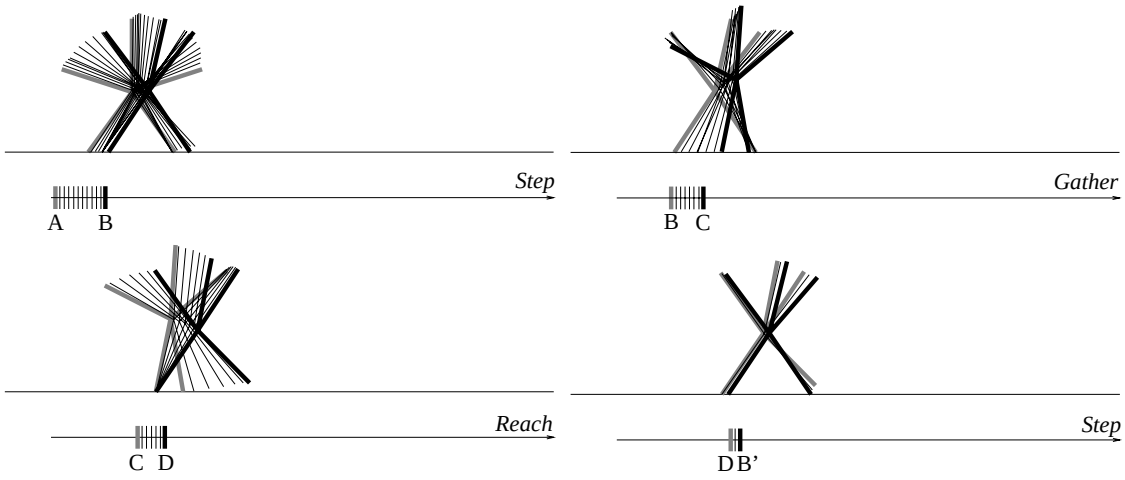


Figure 5: Mr. Star-Man shuffling. The B–C–D–B' sequence is repeated cyclically.

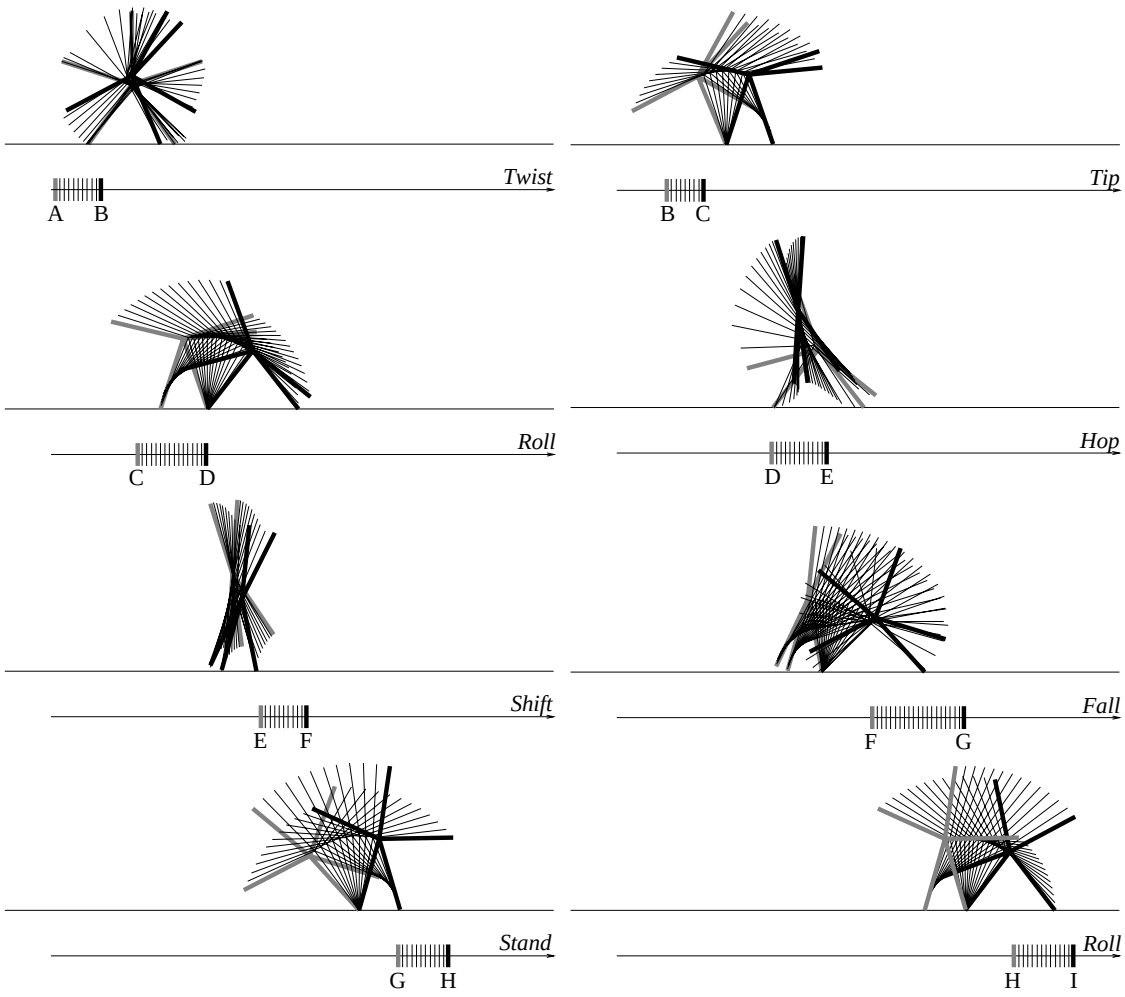


Figure 6: Mr. Star-Man doing a cartwheel.

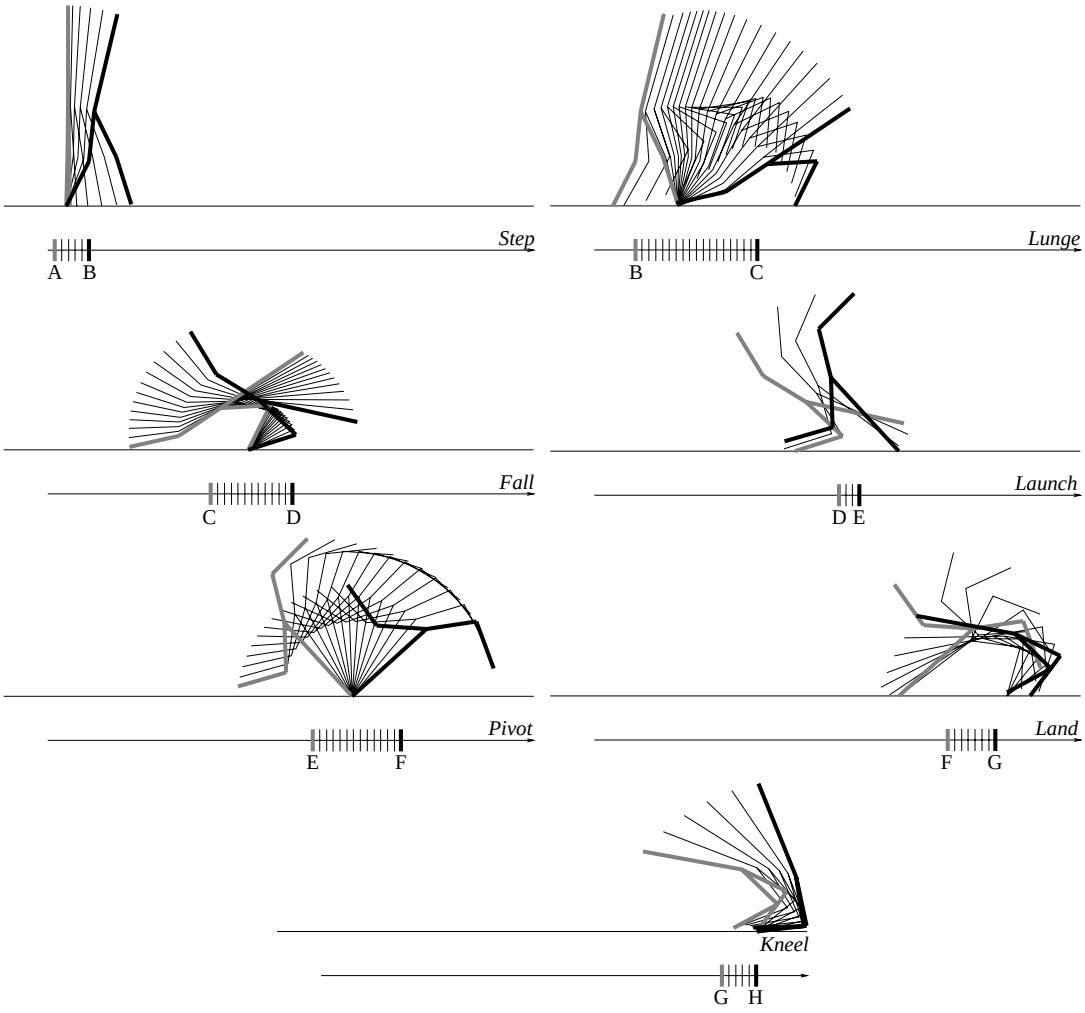


Figure 7: Beryl Biped tumbles aperiodically.

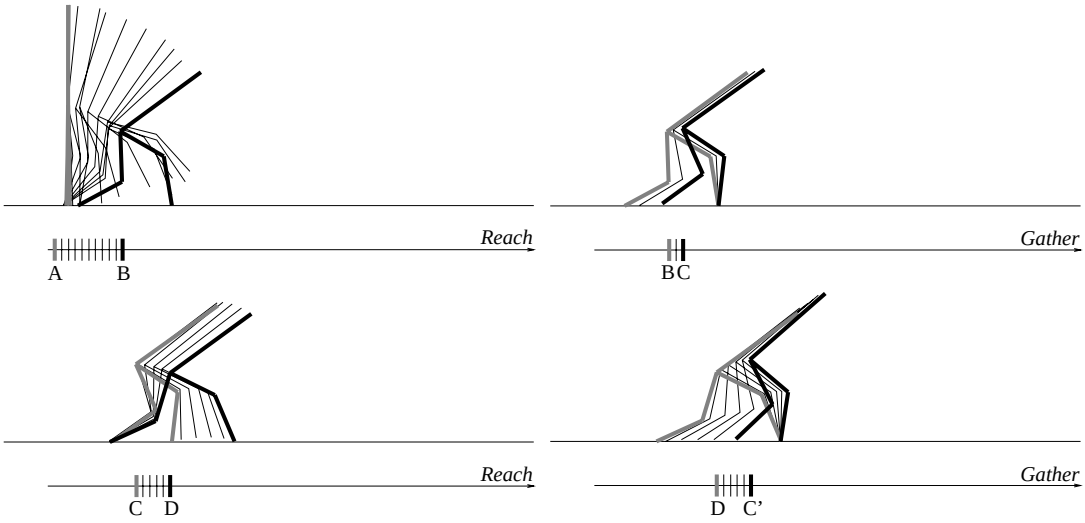


Figure 8: Beryl Biped shuffles periodically. The C–D–C' sequence is repeated cyclically.

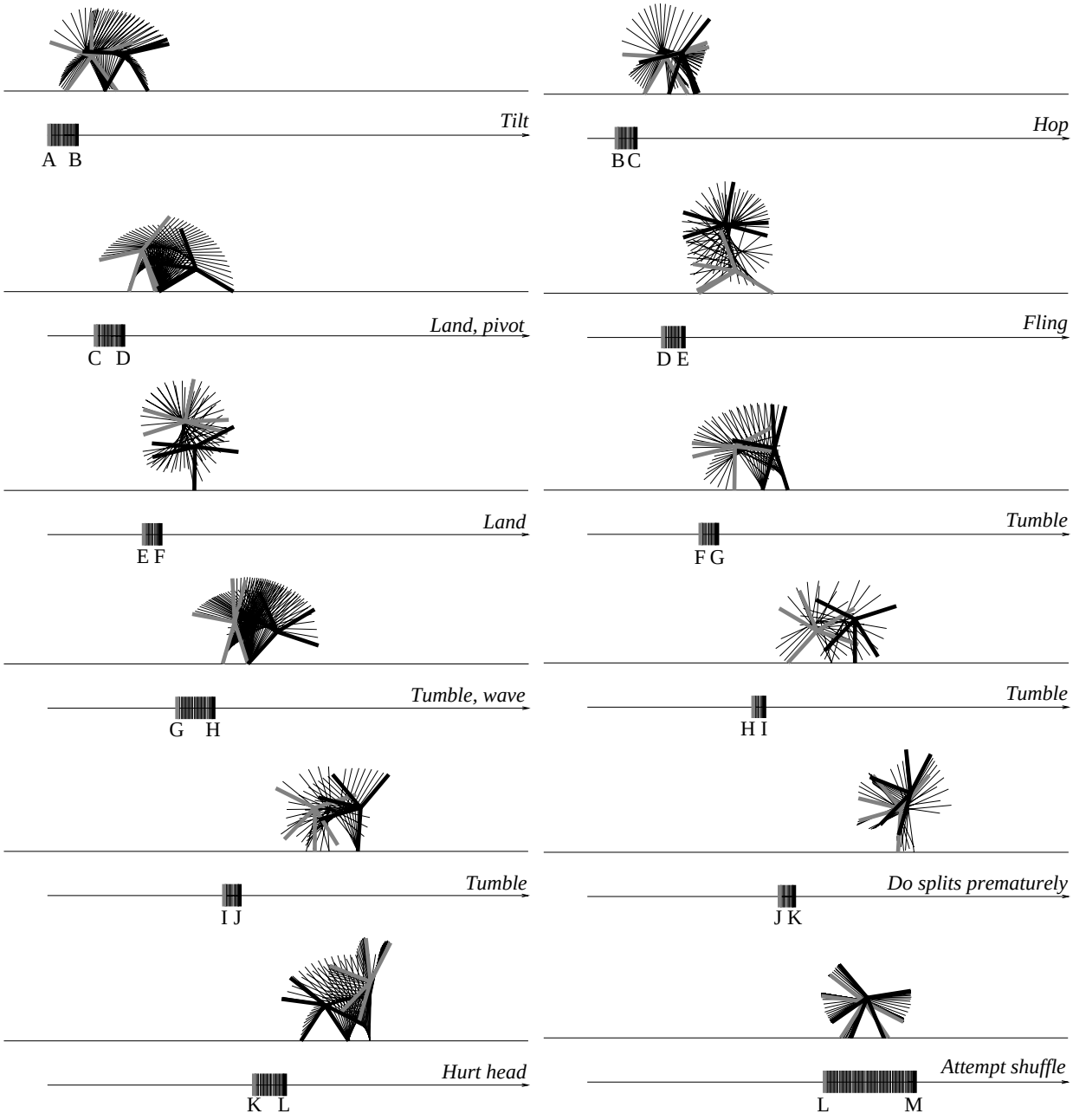


Figure 9: Mr. Star-Man's composite trajectory before refinement.

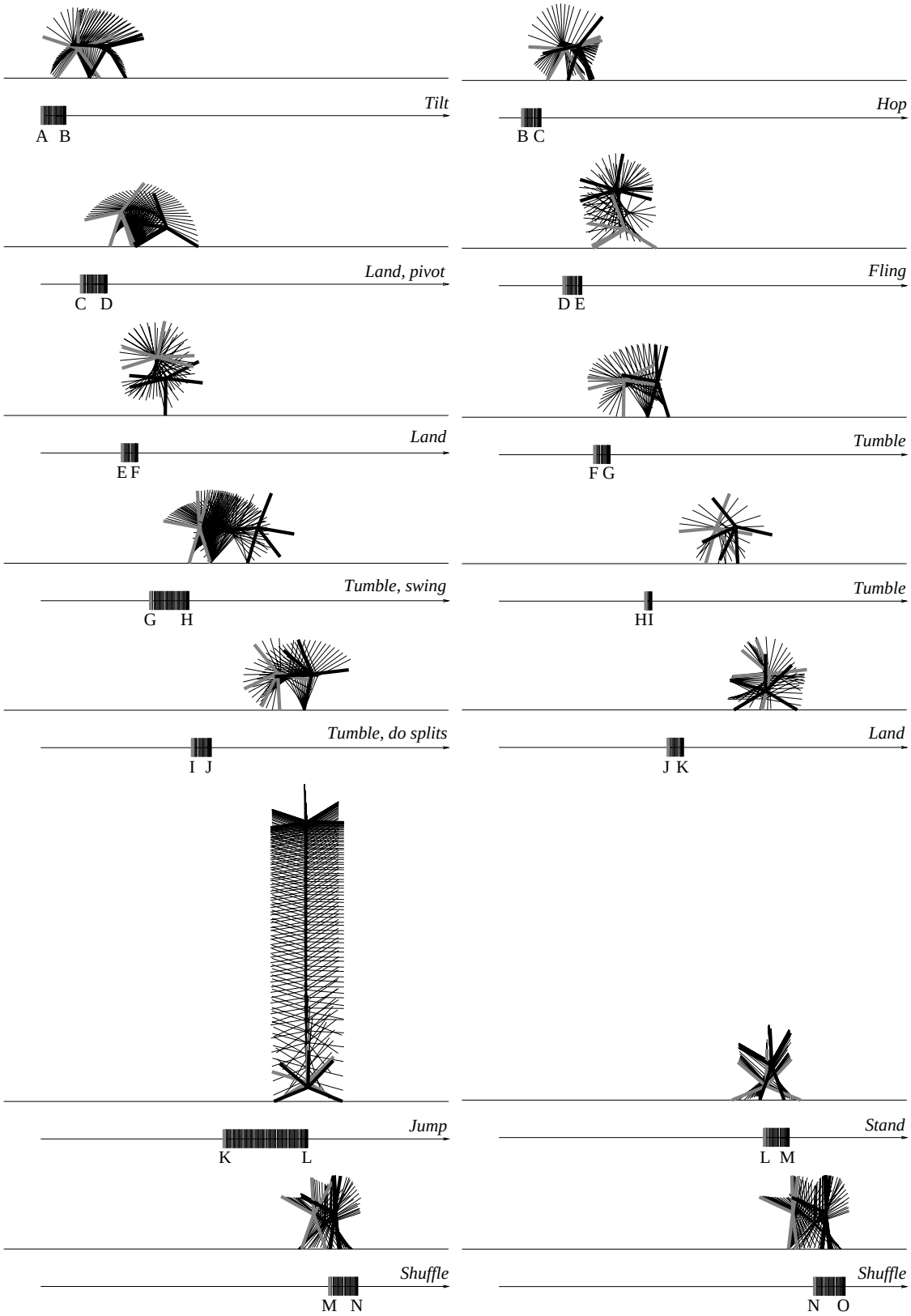


Figure 10: Mr. Star-Man's composite trajectory after refinement.

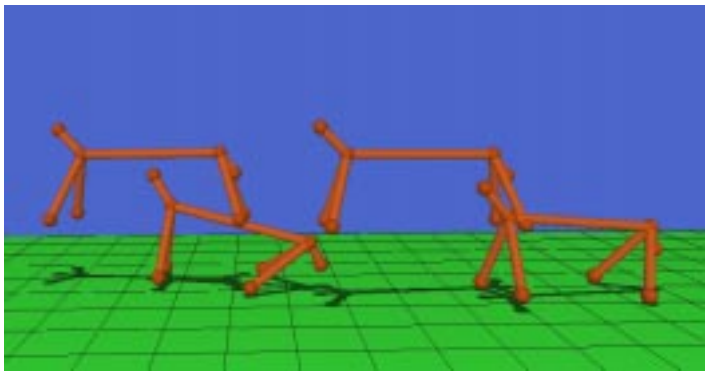


Figure 11: Cujo bounds from right to left.

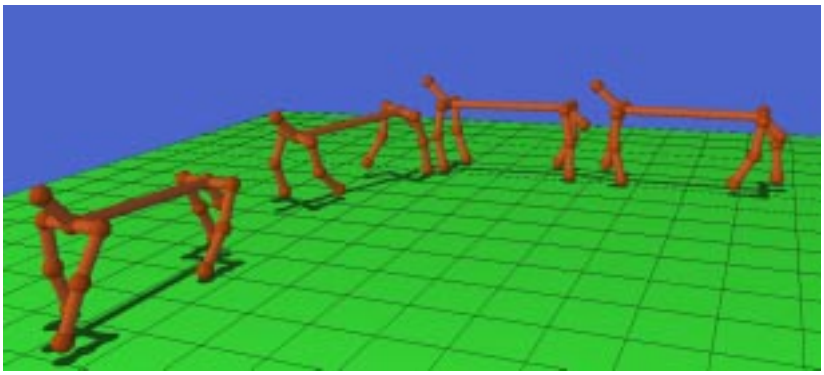


Figure 12: Rex walks, turns, and walks again.

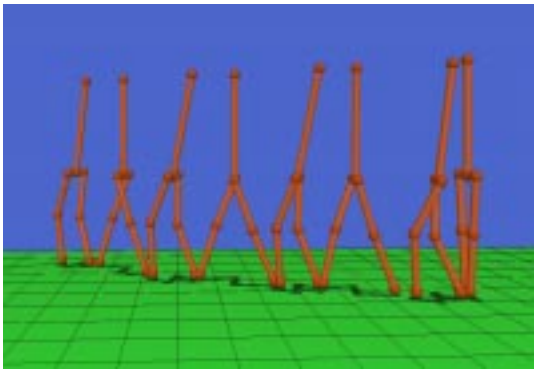


Figure 13: Bob the Biped walks.