Further Experience with Controller-Based Automatic Motion Synthesis for Articulated Figures[*]

Joel Auslander, Harvard Univ.
Alex Fukunaga, UCLA
Hadi Partovi, Harvard Univ.
Jon Christensen, Harvard Univ.
Lloyd Hsu, Harvard Univ.
Peter Reiss, Harvard Univ.
Andrew Shuman, Harvard Univ.
Joe Marks, Mitsubishi Electric Research Labs
J. Thomas Ngo, Interval Research

Abstract

We extend an earlier automatic 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 controller-based approach to automatic motion synthesis for computer animation.

[*]Much of the work described in this paper was done while JM was at Digital Equipment Corporation's Cambridge Research Lab and AF and JTN were at Harvard University. Current author affiliations (where different): JA is at the University of Washington; HP and AS are with Microsoft Corporation; LH is with Oracle Corporation; and PR is with D.E. Shaw & Co. Correspondence should be addressed to Joe Marks at MERL, 201 Broadway, Cambridge, MA 02139, U.S.A. (e-mail:marks@merl.com).

CR Categories: I.2.6 [**Artificial Intelligence**]: Learning—*parameter learning.* I.2.6 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search—*heuristic methods.* I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*animation.* I.6.3 [**Simulation and Modeling**]: Applications.

Additional Key Words: Animation, spacetime constraints, heuristic methods, machine learning, stochastic optimization, evolutionary computation.

1 Introduction

Automatic motion synthesis is the problem of determining how an animated character should move in a visually plausible way that meets the animator's goals (see Figure 1). (In this paper and much of the cited literature, 'movement' is taken to mean locomotion, though more general kinds of motion are also important for animation and should be included under the general rubric of motion synthesis [2, 14].) Animated characters are often modeled as articulated figures, comprising rigid rods connected by flexible joints; this is the kind of physical model we consider here [1]. In this context, the automatic motion-synthesis paradigm requires the animator to specify only the physical structure of an articulated figure and quantitative criteria for success in the desired locomotive task. The computer then finds a physically realistic motion for the figure (physical realism is a good way to guarantee visual plausibility) that is near optimal with respect to the task success criteria.

One approach to automatic motion synthesis uses local optimization to refine initial articulated-figure trajectories by making them more compliant with physical law (the "Spacetime Constraints" approach) [8, 16, 31]. Another form of local optimization involves straightforward use of optimal-control theory to improve the motion with respect to the task criteria [7]. However, local optimization has inherent limitations for this problem: it will usually be confounded by the discontinuities and local optima found in the search space of a typical motion-synthesis problem, thus leaving primary responsibility for constructing coarse initial trajectories with the human animator. While this may be an acceptable, even attractive option for professional animators, users without the requisite artistic skills may find it much less useful.

Recently, an alternative approach to the motion-synthesis problem was proposed. This approach does

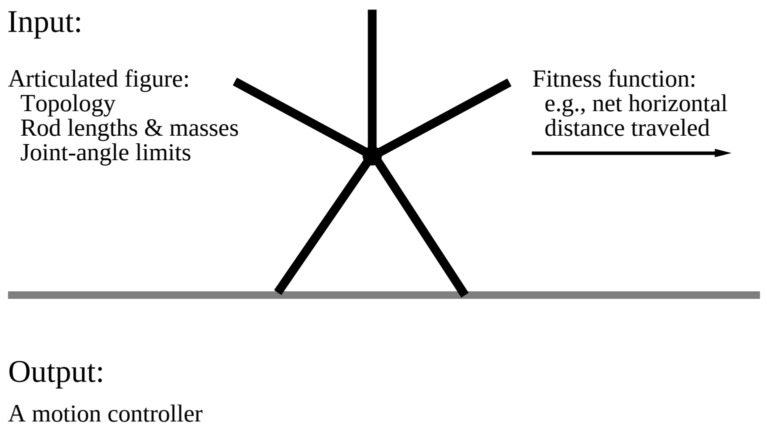


Figure 1: An illustration of controller-based automatic motion synthesis.

not use local optimization to refine a figure's initial trajectory, but instead computes trajectories de novo. This is done by automatically generating a *motion controller* that can actuate the figure to produce the desired motion [19, 20, 24, 25, 26].[1] 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 [24], 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 [26] 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

[1]Before its introduction to the graphics community, various incarnations of what we call controller synthesis had been proposed in the AI, robotics, and optimal-control communities, *e.g.*, [6, 9, 11, 15, 17, 22].

ascent. More recently, Sims [25] described a comparable approach to automatic motion synthesis as part of a system that generates both the structure of articulated figures and controllers that cause them to move in desired ways.

In contrast with these approaches, Ngo and Marks employ a “winner-take-all” controller representation in which the character is governed, at any given instant, by a single very simple *stimulus-response* rule (the “winner”) instead of a superposition of many rules. Which rule is active, and therefore which actuation response is performed, is determined by the current state of the physical environment (the stimulus). Thus these banked stimulus-response (BSR) controllers are similar in form to various state-machine control systems that have been designed by hand to animate articulated figures [13, 23]. The space of possible BSR controllers is searched by an evolutionary-computation strategy [10, 19, 20].

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 generate motions that match an animator’s preconceptions—although near-optimal trajectories are produced, the random component of the search process makes it difficult to predict or influence 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 [8] in the context of the original Spacetime Constraints paradigm [31].)

4. The existing algorithm works for 2D articulated figures only—generalizing to 3D would appear to be computationally formidable.

We show how each of these problems can be addressed to some degree. We begin by summarizing several incremental refinements 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 motion-synthesis 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 [19, 20] 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. [10] demonstrated that a simpler, serial search algorithm, also based on the principles of evolutionary computation, exhibits far better performance. Below we review the details of the physical simulator, BSR controllers, and the serial search algorithm used in previous work.

2.1.1 The physical simulator

The design of any physical simulator for articulated figures represents a compromise between physical accuracy and computational efficiency. Because of the demands of the search process (tens of thousands of physical simulations may be required to find a good BSR controller—see below), efficiency is our principal concern. Following the approach described by Hahn [12], we handle the internal deformations of an articulated figure *kinematically*, without computing the forces and torques required for these deformations; the motion of the figure within its environment is determined *dynamically*, using the laws of physics as they apply to a fully rigid articulated figure.

More specifically, each internal joint angle is governed by a kinematic equation of the following form:

$$\tau^2\ddot{\theta} + 2\tau\dot{\theta} + (\theta - \theta^0) = 0.$$

The values of θ^0 (the joint angles in the *target pose*) and τ (a uniform time constant for the whole figure) come from a BSR controller, described below.[2] At each time step, the figure deforms in accordance with its kinematic equations, and a new moment of inertia is computed for the now rigid figure. Corrections are then made to the figure's position and orientation that compensate for changes in linear and angular momentum caused by the deformation. Finally, a simulation of standard rigid-body forward dynamics is performed that accounts for gravity and the forces and impulses that result from contact and collision with the ground. (Collisions between rods are ignored.) The contact forces and collision impulses are computed using a variant of Baraff's method [3, 4, 5] that accounts for additional contributions to the contact forces that can arise from internal deformation of the figure. We use a very simple friction model: when only one rod endpoint is in contact with the ground, it is not allowed to move horizontally; when two endpoints touch (for 2D articulated figures we assume that only two endpoints can be in contact with the ground simultaneously), the ratio of the frictional forces parallel to the floor is equal to the ratio of the contact forces normal to the floor.

2.1.2 BSR controllers

A BSR controller governs a vector $\vec{\theta}^0(t)$ of target 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.

A controller contains R stimulus-response rules. (The controllers that generated the trajectories illustrated in Figures 9 and 10 [3] each have $R = 10$ rules, though typically only three or four rules contribute significantly to the motion.) Each rule i is specified by stimulus parameters $\vec{S}_i^{\mathrm{lo}}$ and $\vec{S}_i^{\mathrm{hi}}$, and response

[2] One problem with this approach is that an articulated figure may deform in ways that would require the exertion of excessive torque about its joints. This problem is ameliorated by restricting the values of τ to ranges that produce acceptable behavior.
[3] All trajectories depicted in this paper are available for viewing on the World Wide Web at URL http://www.das.harvard.edu/users/faculty/Joe_Marks/animation-poster.html.

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

Table 1: Components of a vector $\vec{S}$ of sense variables for an N-rod articulated figure. All sense-variable values are normalized to lie in the range $[0.0, 1.0]$.

parameters $\vec{\theta}_i^0$ and τ_i. So if N is the number of rods in an articulated figure, $|\vec{S}_i^{\text{lo}}| = |\vec{S}_i^{\text{hi}}| = 2N + 2$ (see Table 1), $|\vec{\theta}_i^0| = N - 1$, and the total number of free parameters in a BSR controller is therefore $(5N + 4)R$.

Based on the instantaneous value of the sense vector $\vec{S}(t)$, exactly one rule is active at any one time. More specifically, each rule i receives a score based on how far the instantaneous sense vector $\vec{S}(t)$ falls within the hyperrectangle whose corners are $\vec{S}_i^{\text{lo}}$ and $\vec{S}_i^{\text{hi}}$. This score is given by the expression:

$$W \cdot \left(1 - \max_{j=1}^{V} \left[\lambda_j(\vec{S}(t)[j] - \mu_j)\right]^2\right),$$

where V is the number of sense variables, $\vec{S}(t)[j]$ is the j^{th} sense variable in $\vec{S}(t)$, $\mu_j = (\vec{S}_i^{\text{hi}}[j] + \vec{S}_i^{\text{lo}}[j])/2$, $\lambda_j = 2/(\vec{S}_i^{\text{hi}}[j] - \vec{S}_i^{\text{lo}}[j])$, and the weight W is:

$$W = \sum_{j=1}^{V} \log\left(\lambda_j / \lambda_j^{\text{min}}\right),$$

with λ_j^{min} being the smallest allowable value of λ_j. Thus the locus of points for which this expression is positive is a hyperrectangle with side lengths $(2/\lambda_1, 2/\lambda_2, \ldots, 2/\lambda_V)$ centered at $(\mu_1, \mu_2, \ldots, \mu_V)$, and rules with smaller, more specific hyperrectangles are favored by the scoring function. The rule with the highest positive score is said to be *active*. If no rule has a positive score, the rule active in the previous time step remains active. The active rule is then used to determine a target pose for the articulated figure. The pseudocode in Figure 2 summarizes how a BSR controller behaves and is evaluated.

2.1.3 The search algorithm

The search algorithm in Figure 3 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.

```
Set $i_{\text{active}}$ to 1
for $t = 0$ to $T - \Delta t$ in increments of $\Delta t$ do
  Kinematic deformation: cause joint angles $\vec{\theta}(t + \Delta t)$
    to approach $\vec{\theta}^0_{i_{\text{active}}}$ with time constant $\tau_{i_{\text{active}}}$
  Dynamics simulation: simulate motion for time interval $[t, t + \Delta t]$
  Measure sense variables $\vec{S}(t + \Delta t)$
  Possibly change $i_{\text{active}}$, based on $\vec{S}(t + \Delta t)$
end for
Assign the controller a fitness value based on how well
  the simulated motion meets the animator-supplied task criteria
```

Figure 2: Operation of a BSR controller.

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. For the 2D articulated-figure motions illustrated in Figures 9 and 10, 40,000 BSR controllers were evaluated, taking 3–6 minutes on a single DEC 3000/400 AXP workstation.[4] Physical simulation accounts for the bulk of the running time.

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 initialization process and the mutation operator are also important for efficient searching. The distributions we used for the initial random generation of parameters are summarized in Table 2. For each mutation, local optimization is facilitated by subjecting one rule in the BSR controller to *creep* (*i.e.*, each free parameter in the rule is perturbed slightly), and global optimization is facilitated by reinitializing one rule from scratch. The latter operation differs from the original initialization step in one crucial way: to increase the likelihood that the reinitialized rule will be relevant (*i.e.*, will be active at some point), the new hyperrectangle is located so that one of its corners lies on the locus $\vec{S}(t)$ in the space of sense variables.

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

[4]Ngo and Marks originally reported running times of 30–60 minutes on a CM-2 Connection Machine to compute comparable BSR motion controllers [19, 20]. Van de Panne and Fiume reported running times of 1–6 hours on a Sun SPARC IPC for their algorithm [26].

```
Initialize a set of $C$ random motion controllers
Evaluate each motion controller in the set
**for** i = 1 **to** $G$ **do**
  **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 if**
  **end for**
  **if** (i **mod** $R$) = 0 **then**
    Rank order the set of controllers
    Replace bottom 50% of the set with top 50%
  **end if**
**end for**
Return the best controller

Reasonable parameter values:
  $C = 10$
  $G = 4{,}000$
  $R = 200$
  Total number of physical simulations = $C \times G = 40,000$
```

Figure 3: Serial search algorithm.

Symbol	Description	Lo	Hi	Distribution
τ	Time constant	$2\Delta t$	$T/4$	Logarithmic
θ^0	Joint angle in a target pose	$\theta^{\min}$	$\theta^{\max}$	Uniform
μ	Hyperrectangle-center coordinate	-0.5	1.5	Uniform
$2/\lambda$	Hyperrectangle side length	0.4	4	Logarithmic

Table 2: The probability distributions used to generate the initial parameter values in an SR rule.

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

Table 3: Secondary terms in the fitness function.

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 9) and a cartwheel (Figure 10). 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 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 and are given lesser weights, 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 3.

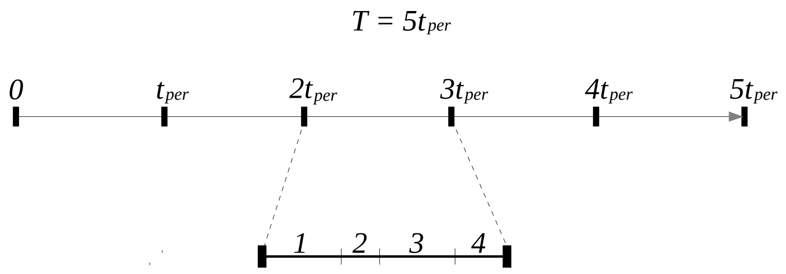
For the sample problem we proposed above, assigning a positive weight only to the secondary term `contact_distance` guarantees a shuffling gait, whereas assigning a positive weight only to the secondary term `rotations` guarantees a cartwheel. For our other animated characters, varying the coefficients of the small set of terms in Table 3 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_height` biases the search in favor of hopping or jumping motions, a positive coefficient for `contact_distance` 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 qualitatively different from those produced by SANs, which tend to exhibit characteristic sinusoidal vibrations [21, 27]. 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.[5] (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 4), 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 SR 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 $t_{\mathrm{per}} < T$, then the sequence of actions specified by the rules is repeated periodically. If, on the other hand, $t_{\mathrm{per}} \geq T$, then the action sequence is not constrained to be periodic.

[5] A functionally equivalent form of motion controller has also been described by van de Panne et al. [30].



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

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

In recent papers [19, 20, 26] it has been argued or assumed that good closed-loop controllers with access to information about physical state should be more effective (*i.e.*, produce better, more robust motions) and easier to find than good open-loop controllers based on time alone. Unexpectedly, we found that time-based motion controllers are quite effective for 2D articulated figures, and the space of time-based motion controllers is easier to search than the space of sense-based controllers.[6] (The issue is more complex for 3D articulated figures, as we describe in §4.) Furthermore, by manually varying the value of the period t_{per}, 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 11 was computed with $t_{\text{per}} = T$, where T is the length of time available for the entire motion. The periodic shuffling in Figure 12 was computed with a value of $t_{\text{per}} \ll T$.

[6] Good time-based controllers for the problems we considered can be found in a few tens of seconds on a DEC 3000/400 AXP workstation using the serial search algorithm described in Figure 3. 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 ($N + 1$ for an N-rod articulated figure), and the relatively small number of rules per controller that are needed for effective motion (typically $R = 4$). Because all of the rules in a time-based controller are guaranteed to be active at one time or another, fewer rules are needed than in comparable sense-based controllers, in which most of the rules either never become active, or are not active long enough to have a noticeable effect (see §2.1.2).

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 [31], Cohen [8] implemented a system that permits a human animator to design and refine a trajectory by interactively submitting simple motion-synthesis 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 motion-synthesis problems in times ranging from minutes to seconds on a modern workstation. However, such speedy turnaround is not likely for 3D motion-synthesis 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:[7]

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 3) 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.

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 motion-synthesis 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?

[7]This regimen is very similar in spirit to one described previously by van de Panne et al. [28], though the motion-controller representations in both schemes are very different, and controllers can be concatenated automatically in our approach. Indeed, the broad idea of concatenating motion controllers was proposed and investigated previously in a variety of contexts [1, 2]; our contribution is distinguished by the method we use for concatenating controllers automatically and the scope and generality of the motion controllers considered.

One possibility is that this is not a problem, due to the robustness intrinsic 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 simple interface in which the time duration for which each precomputed controller is executed is specified by a slider bar. 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.

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 13. 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.

As an alternative to this sometimes tedious manual process of selecting transition points, we implemented a semiautomatic system comprising three techniques:

- *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. This is because a controller survives the evolutionary process only to the degree that it is insensitive to the initial physical configuration. (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 array 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 4.
- *A heuristic search strategy:* Beginning with an optional user-supplied guess as to the most appropriate transition points, different transition-point arrays are explored using the search heuristic described in Figure 5.

Thus in this approach the animator is responsible for entering an optional 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 4 and the initial concatenation of motion controllers depicted in Figure 13, our technique adjusted the transition points automatically in under a minute on a DEC 3000/400 AXP workstation to achieve the desired composite motion (consisting of a cartwheel, a jump,[8] and a shuffle) depicted in Figure 14.

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 usefully 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 (see Figure 6). More importantly, for many 3D articulated figures a large fraction of the search

[8]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."

```
if user-supplied transition points are available then
  Generate a set of $Q$ identical transition-point arrays,
    all duplicates of the user-supplied input
else
  Generate a set of $Q$ identical default transition-point arrays
    (with transition points that are equally spaced along the time interval)
end if
Evaluate each transition-point array in the set
for i = 1 to $G$ do
  for each individual transition-point array in the set do
    for j = 1 to $H$ do
      Mutate (i.e., randomize $\frac{1}{3}$ of the time, perturb $\frac{2}{3}$ of the time)
        a randomly chosen transition point
      Evaluate the new transition-point array
      if the new array is better than the old one then
        Replace the old array with the new one
      end if
    end for
  end for
  Rank order the set of transition-point arrays
  Replace bottom 50% of the set with top 50%
end for
Return the best transition-point array

Reasonable parameter values:
  $Q = 16$
  $G = 16$
  $H = 4$
  Total number of physical simulations $= Q \times G \times H = 1,024$
```

Figure 5: Search heuristic for composite motion synthesis.

Script:

$$\begin{array}{lcl} \mathit{cartwheel} & = & \Delta cm_x \times (\Delta time \overset{?}{=} 159) \\ \mathit{jump} & = & \max_t\{\min_i\{y_i(t)\}\} \times (\Delta time \overset{?}{=} 90) \\ \mathit{shuffle} & = & \Delta cm_x \times (\Delta time \overset{?}{=} 100) \\ \mathit{total} & = & (1.0 + \mathit{cartwheel}) \times (1.0 + \mathit{jump}) \times (1.0 + \mathit{shuffle}) \end{array}$$

Translation:

- Credit for the cartwheel phase is proportional to the horizontal distance traveled by the center of mass (Δcm_x) times a factor that penalizes deviation of the cartwheel's duration ($\Delta time$) from the original user-supplied duration (159 simulator ticks). (The $\overset{?}{=}$ binary operator returns 1.0 when the two operands are equal, and decreases to 0.0 as they differ.)
- Credit for the jump is a function of the highest height cleared during this phase ($\max_t\{\min_i\{y_i(t)\}\}$) times a factor that penalizes deviation of the jump's duration ($\Delta time$) from the original duration (90 simulator ticks).
- Credit for the shuffle phase is proportional to the horizontal distance traveled by the center of mass (Δcm_x) times a factor that penalizes deviation of the shuffle's duration ($\Delta time$) from the original duration (100 simulator ticks).
- The total value is the multiplicative product shown.

Table 4: A sample composite-motion script.

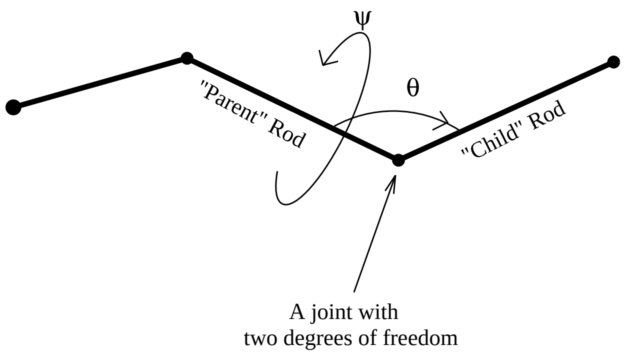


Figure 6: The degrees of freedom in a joint of a 3D articulated figure.

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 4), 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 angles for all the joints in the articulated figure. Thus, a time-based BSR controller comprising R rules contains $R-1$ independent variables to divide up each period of time t_{per}, $(2N-2)R$ target joint 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 16) has $N = 15$ rods and $R = 4$ SR rules in his motion controller, so the total number of variables is 119. Some of these variables happen to be fixed in Rex (about two thirds of the joint angles are fixed; the remaining one third have an average range of 40°), so the search algorithm must find 47 floating-point variable values 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 3.)

Figure 15 shows a trajectory that is typical of those produced by time-based BSR motion controllers. Cujo, a dog-like creature having seven rods and an average joint-angle range of 25°, propels himself forward by bounding repeatedly. A trajectory for Rex, depicted in Figure 16, is the result of separate motion-synthesis problems that were solved seriatim (*i.e.*, using an approach analogous to Cohen's [8]): 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 (except for the turning motion, which measured the change in heading), 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 7: 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 3D 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 open-loop time-based BSR motion controllers make them very attractive when compared to the original closed-loop 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-

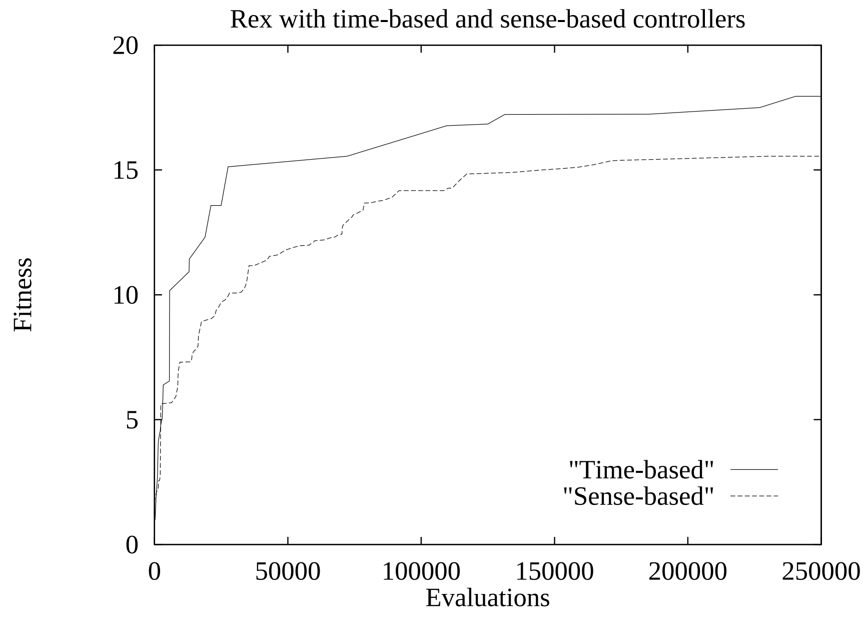


Figure 7: Learning curves for both kinds of motion controller.

synthesis problem we have considered is that of making Bob the Biped walk. Bob, a biped with point feet (see Figure 17), is very unstable, so he falls over easily. (Bob comprises nine rods, and his joints have the flexibility one would expect of a humanoid, so roughly one third of the joint angles are fixed, and the remaining two thirds have an average range of a little less than 25°.) 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.[9]

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 angles for each flexible joint. The increased complexity of each SR rule and the desirability of having several rules (typically 10) in a sense-based BSR controller means that 880 floating-point numbers are used to specify a

[9]Bob's point feet make his motion-synthesis task akin 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 be difficult to design. However, the additional computational cost of a more general physical simulator able to handle this might place results comparable to those reported here beyond the reach of current serial hardware.

sense-based motion controller for Bob the Biped. (The corresponding number for Rex (Figure 16) is 1,280.) The basic form of the search algorithm remains the same (Figure 3), but the initialization and mutation procedures require modification in order to find good walking controllers for Bob. Recall that the addition of a randomly generated SR rule to an existing sense-based controller is unlikely to affect the generated motion, because a randomly generated stimulus hyperrectangle occupies a very small fraction of the full volume of the space of sense variables; and the greater the number of sense variables, the worse this problem becomes. 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) play a useful role, it is necessary to employ a *relevance heuristic*, which in this context is a method for generating random SR rules that are guaranteed to affect the generated motion. The relevance heuristic employed in the original work on BSR controllers for 2D articulated figures (§2.1.3) 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.

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 7), they were distinctly superior for Bob the Biped. The walk depicted in Figure 17 was one of two effective, forward-facing walking strategies discovered.[10] (The strategy not shown here is a highly periodic shuffling motion.) We also noticed a considerable variation in the scores of the sense-based BSR controllers found by different runs of the same algorithm (see Figure 8), 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

[10]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.) It should also be noted that although the resulting controllers produced effective ambulation for the duration of the simulation, they are probably not stable.

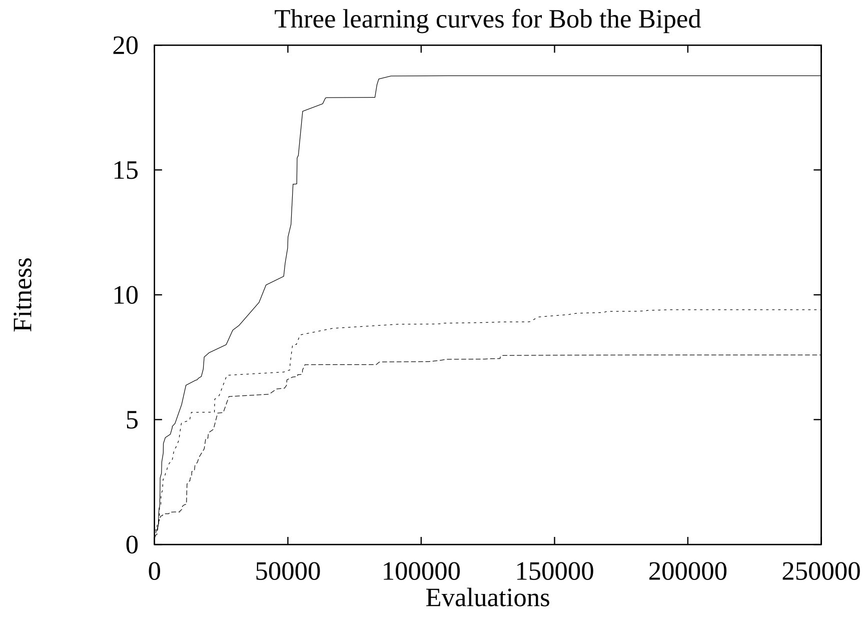


Figure 8: Variability in search-algorithm performance.

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 serve 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 [18, 19, 20], a renewed investigation of parallel search for motion synthesis may be worthwhile. An even better idea might be to exploit whatever help the animator can supply to expedite the search process, or to use a different physical model (*e.g.*, mass-spring lattices) that may be easier to simulate and control.

- *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 motion. Unfortunately, they have two major drawbacks: they are not easily concatenated to give composite motions,[11] 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 3 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, precomputed BSR motion controllers, and we have described one way to 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.

- *Is BSR-style motion synthesis useful for nonlocomotive motion tasks?* Although locomotion has been the focus of our work, motion synthesis for animation encompasses nonlocomotive tasks as well. We have been successful in applying our approach to some hybrid motion tasks, such as having an articulated figure reach for and touch a ball in flight. However, it seems clear that other approaches will be

[11] However, van de Panne et al. have recently reported some positive experiences with modifying and concatenating time-based controllers [29].

more useful for more involved manipulation tasks [14].

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 (NSF grant IRI 93-50192).

References

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

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

[3] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3):223–231, July 1989.

[4] D. Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4):31–40, July 1991.

[5] D. Baraff. Issues in computing forces for non-penetrating rigid bodies. *Algorithmica*, 10(2/3/4):292–352, August/September/October 1993.

[6] R. D. Beer and J. C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122, Summer 1992.

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

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

[9] H. de Garis. Genetic programming: Building artificial nervous systems using genetically programmed neural network modules. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 132–139, Austin, Texas, June 1990.

[10] 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)*, pages 76–83, San Diego, California, February 1994.

[11] C. J. Goh and K. L. Teo. Control parameterization: A unified approach to optimal control problems with general constraints. *Automatica*, 24(1):3–18, 1988.

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

[13] J. K. Hodgins, P. K. Sweeney, and D. G. Lawrence. Generating natural-looking motion for computer animation. In *Proceedings of Graphics Interface '92*, pages 265–272, Vancouver, British Columbia, May 1992.

[14] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe. Planning motions with intentions. In *SIGGRAPH '94 Conference Proceedings*, pages 395–408, Orlando, Florida, July 1994. ACM SIGGRAPH.

[15] J. R. Koza and J. P. Rice. Automatic programming of robots using genetic programming. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 194–201, San Jose, California, July 1992. American Association for Artificial Intelligence.

[16] Z. Liu, S. J. Gortler, and M. F. Cohen. Hierarchical spacetime control. In *SIGGRAPH '94 Conference Proceedings*, pages 35–42, Orlando, Florida, July 1994. ACM SIGGRAPH.

[17] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 796–802, Boston, Massachusetts, 1990. American Association for Artificial Intelligence.

[18] 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.

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

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

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

[22] M. G. Pandy, F. C. Anderson, and D. G. Hull. A parameter optimization approach for the optimal control of large-scale musculoskeletal systems. *Transactions of the ASME: Journal of Biomechanical Engineering*, 114:450–459, November 1992.

[23] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics*, 25(4):349–358, July 1991.

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

[25] K. Sims. Evolving virtual creatures. In *SIGGRAPH '94 Conference Proceedings*, pages 15–22, Orlando, Florida, July 1994. ACM SIGGRAPH.

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

[27] 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.

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

[29] M. van de Panne, R. Kim, and E. Fiume. Synthesizing parameterized motions. In *Proceedings of the Fifth Eurographics Workshop on Animation and Simulation*, Oslo, Norway, September 1994.

[30] M. van de Panne, R. Kim, and E. Fiume. Virtual wind-up toys for animation. In *Proceedings of Graphics Interface '94*, pages 208–215, Banff, Alberta, May 1994.

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

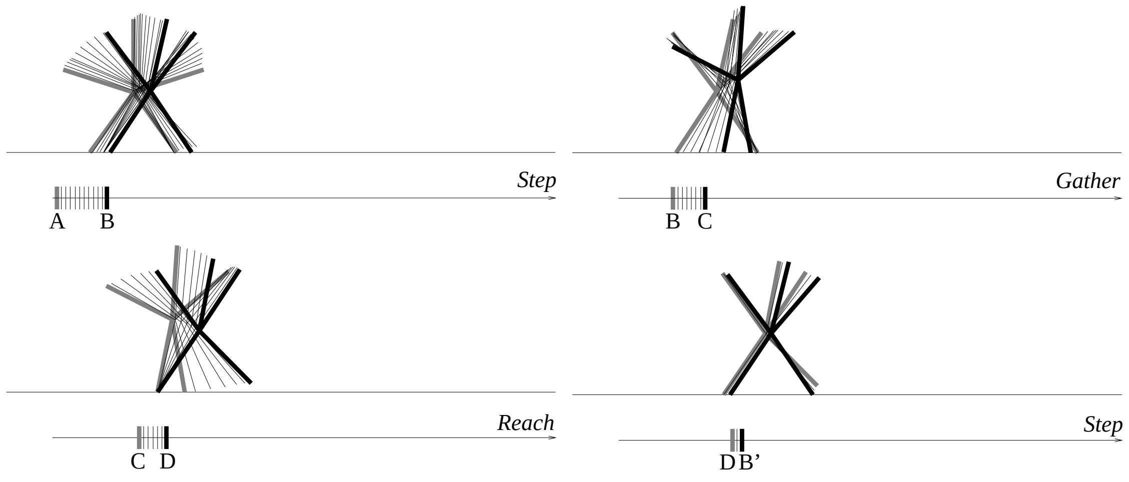


Figure 9: Mr. Star-Man shuffling. The B–C–D–B' sequence is repeated cyclically.

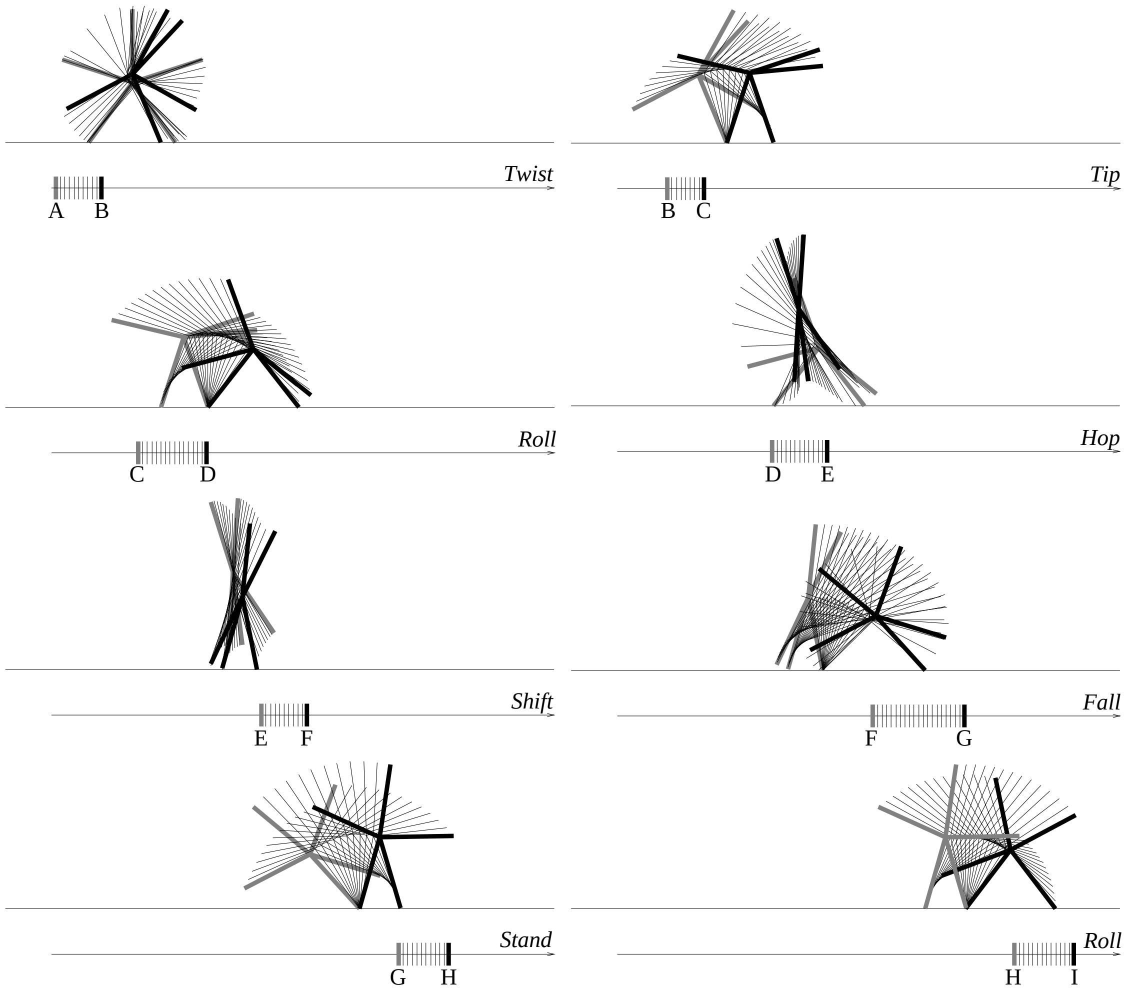


Figure 10: Mr. Star-Man doing a cartwheel.

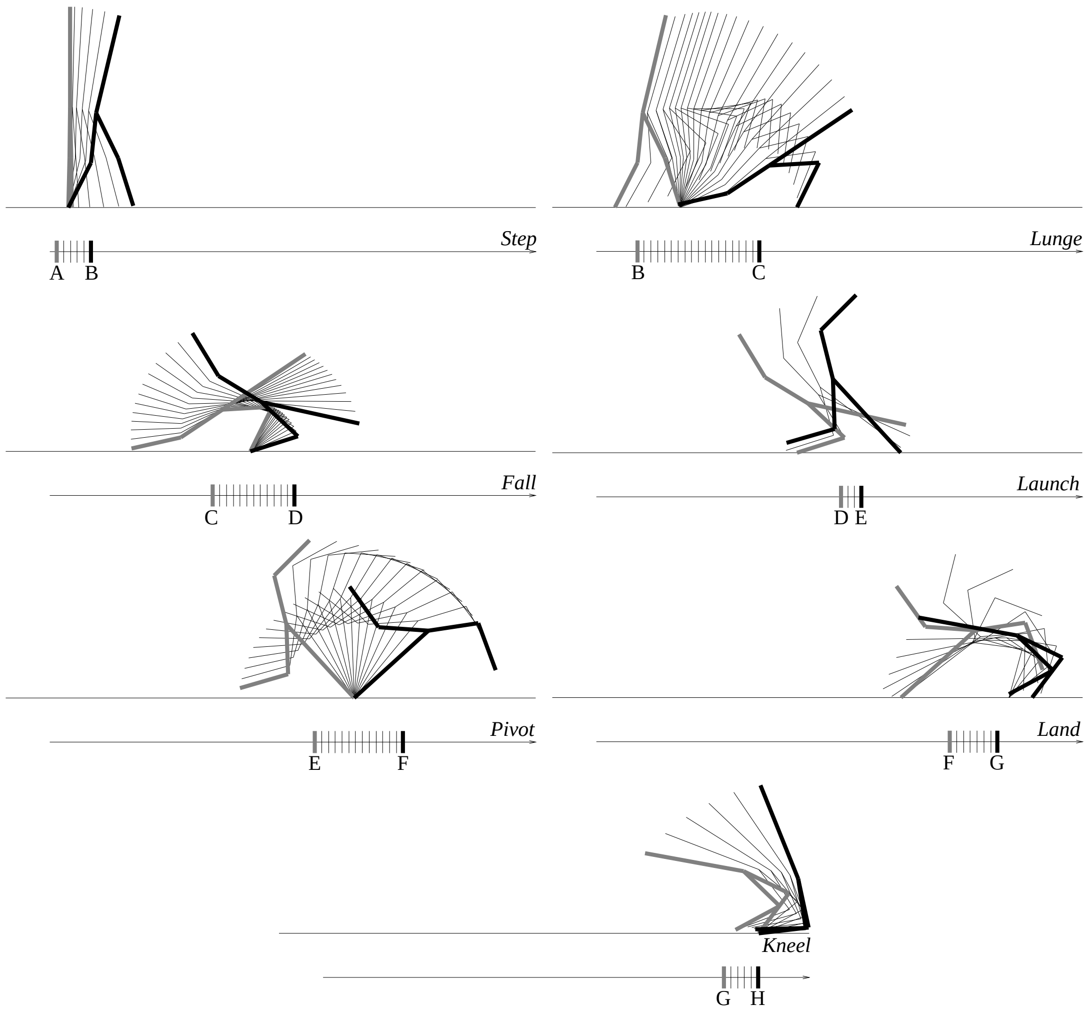


Figure 11: Beryl Biped tumbles aperiodically.

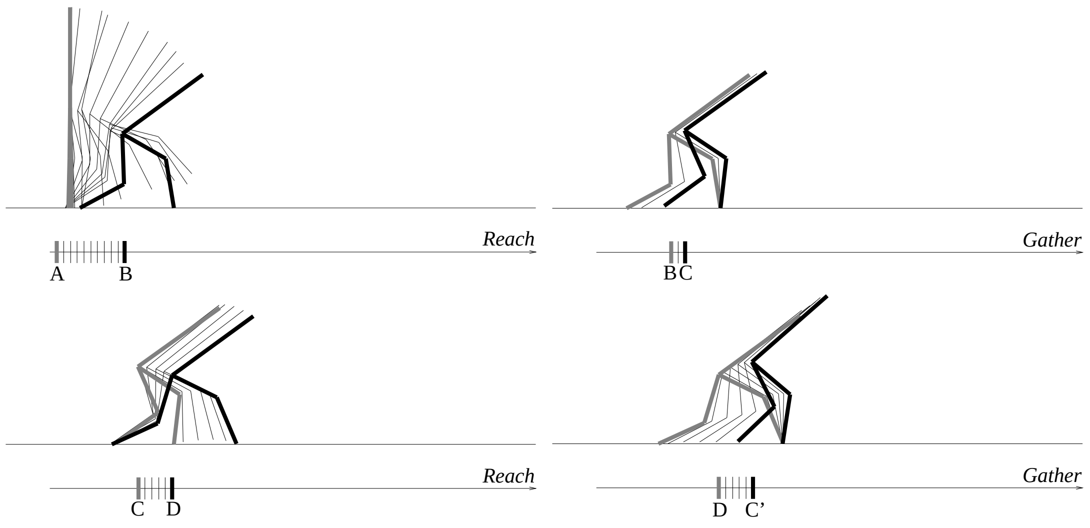


Figure 12: Beryl Biped shuffles periodically. The C–D–C' sequence is repeated cyclically.

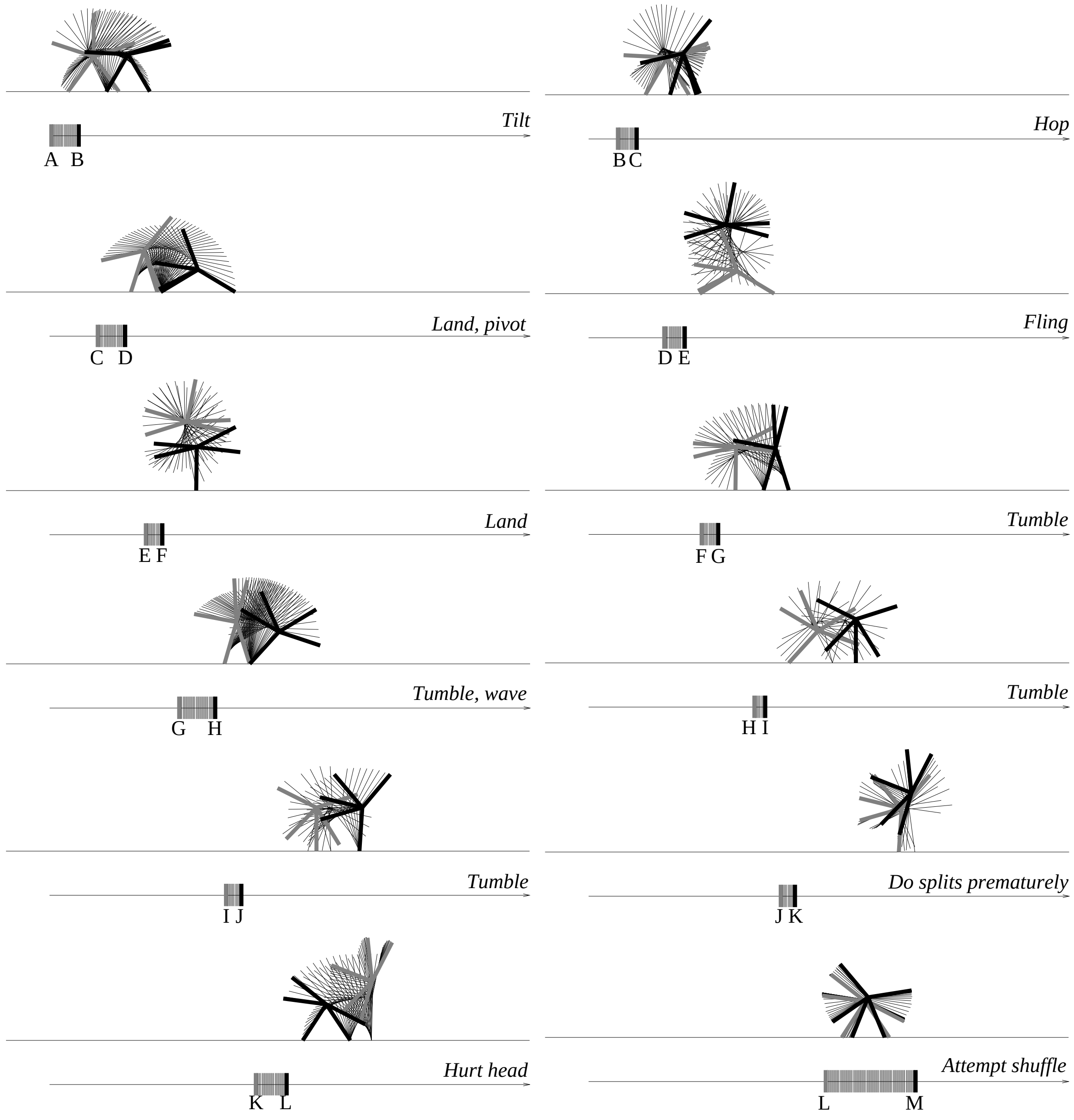


Figure 13: Mr. Star-Man's composite trajectory before refinement.

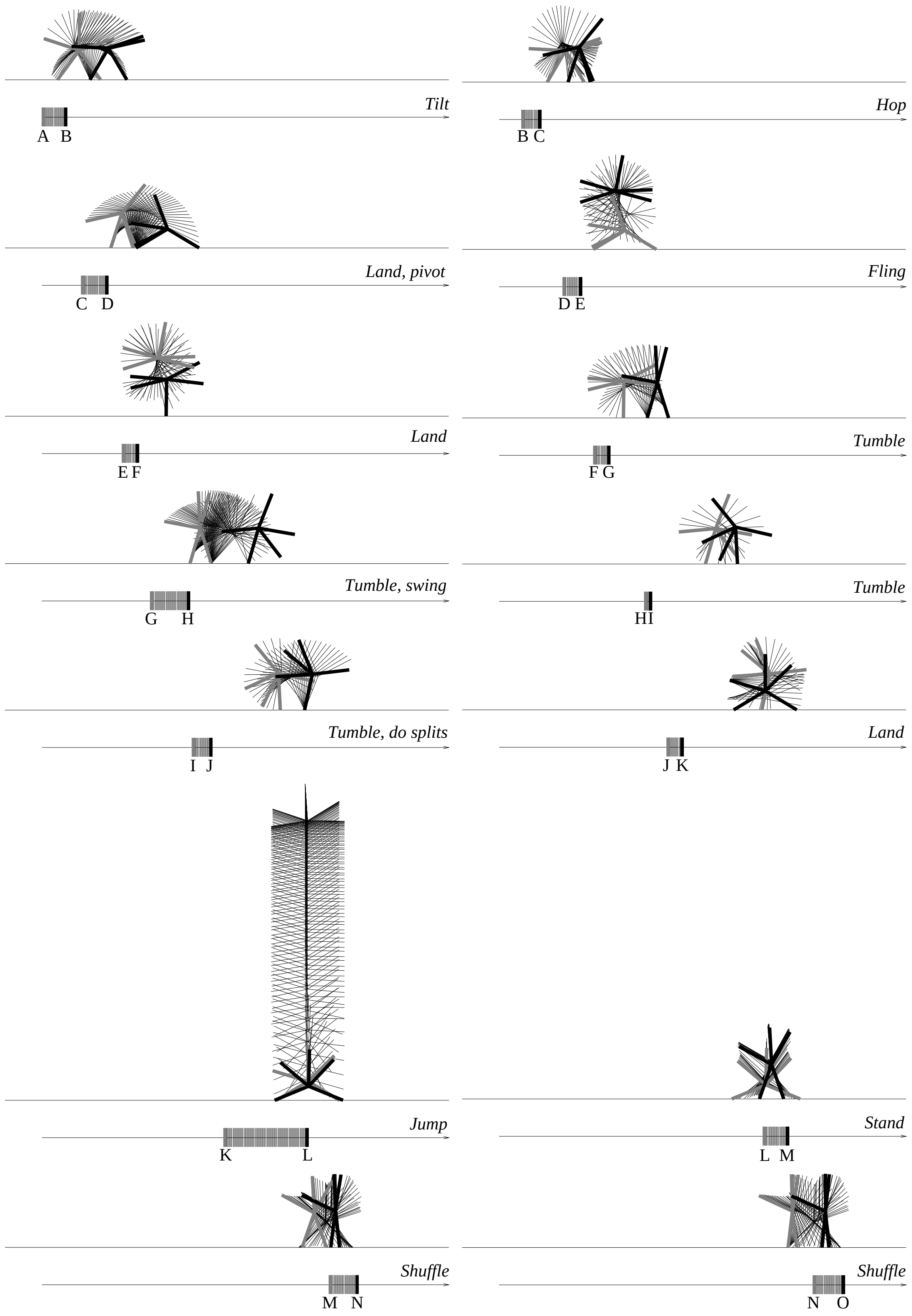


Figure 14: Mr. Star-Man's composite trajectory after refinement.