[to be presented at the Fifth Eurographics
Workshop on Animation and Simulation, Sept. 17-18, 1994]

Synthesizing Parameterized Motions

Michiel van de Panne
*Ryan Kim**
Eugene Fiume

Dept. of Computer Science
*Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Canada, M5S 1A4
{van | rkim | elf}@dgp.utoronto.ca

Abstract

In striving to construct higher level control representations for simulated characters or creatures, one must seek flexible control representations to build upon. We present a method for the synthesis of parameterized, physics-based motions. The method can be applied to both periodic and aperiodic motions. The basis of the method is a low-level control representation in which linear combinations of controllers generally produce predictable in-between motions.

1.0 Introduction

Many of the most interesting objects to animate, such as humans, animals, and robots, are capable of controlling their own motion using muscles or actuators. While realistic motion can be obtained for these *active systems*[15] by applying the Newtonian laws of physics, this also requires solving an associated control problem. Informally stated control problems for animation such as "jump from A to B, then walk to the left" can be posed in a variety of ways. In this paper we examine how to produce control solutions that are reusable and can be parameterized. Producing control solutions to a class of motions as opposed to a specific motion eliminates the need to resolve the control problem each time a new variation of a motion is required.

Since the first use of physics-based animation for articulated figures by Wilhelms[23] and Armstrong and Green[1], a variety of control techniques have been proposed. Some of these draw upon control theory or biological motor control, while others focus directly on creating a usable tool for animation. The approaches taken by this latter group can be broadly divided into two classes. The first poses the problem in terms of a trajectory through state-space and time which is subject to the constraints of physics and the constraints of the desired motion. The second approach involves creating a controller which produces motion by directly supplying actuating forces and torques to a mechanical simulation. The first approach has closer ties to keyframing, while the second better reflects the way movement is generated in real humans, animals, and robots.

The trajectory-based approach iteratively modifies an initial trajectory towards satisfying the laws of physics and user-imposed constraints while optimizing a user-defined objective function. This method was originally proposed by Witkin and Kass[24] and subsequently extended by Cohen[4]. A problem (or feature) of this method is that the laws of physics are treated as a soft constraint, thus the resulting motion is not guaranteed to be physically plausible.

We shall focus on the alternative approach to solving the control problem, namely creating controllers. Controllers provide the benefit of capturing a skill in a reusable way. A controller can be made even more versatile by parameterizing it in a desired way. A simple example is that of producing gaits of different speeds using a single controller. Lastly, controllers can make use of feedback, which often allows for a greatly simplified representation of the control and provides the capability to take corrective actions while a motion is in progress.

Many control structures used in the context of animation take the form of finite-state machines. Zeltzer uses a hierarchy of finite-state machines to provide kinematic control over a human skeleton[25]. Hodgins, Sweeney, and Lawrence use such a representation to control juggling, pumping a swing, and riding a see-saw[7]. Stewart and Cremer use a state-machine to add and remove constraints in order to control a walking biped[18]. The kinematic walking of Bruderlin and Calvert[3] could also be considered as being driven by a type of state machine. Most of these systems are governed by controllers that discretise a periodic cycle into several states. An alternative is to make direct use of sinusoidal oscillations, as done by Miller[12], McKenna and Zeltzer[11], and van de Panne, Fiume, and Vranesic[21]. The work of Raibert and Hodgins[15] in controlling hopping and running motions provides an important step forward, showing how an effective decomposition can be used for this class of motions to yield a set of simpler, solvable control problems.

The work of Ngo and Marks[13] and van de Panne and Fiume[20] introduced methods of automating the synthesis of controllers for arbitrary types of simple articulated figures. The control structures used in these methods can also be considered to be finite-state machines. Each state specifies a desired pose and transitions between states are either timed or based upon sensory data. In neither case was it shown how the synthesized controllers could be scripted or parameterized. The search techniques used in these papers differ in form but are similar in function. Ngo and Marks sample the parameter space in a global way by initially creating a random population of controllers. Local searches are then performed using the mutation and crossover operations of genetic algorithms. Van de Panne and Fiume explicitly separate the global and local search of the parameter space.

The work in progress reported upon here shows how finite-state machines may be automatically synthesized using the two-phase optimization technique reported in [20]. We name our control representation *pose-control graphs* because each state specifies a desired internal *pose* for the creature to take. In their simplest form, namely cyclic pose control graphs, they operate as virtual wind-up toys. A study of *cyclic* pose-control graphs and an analysis of the motions produced is given in [22]. We use the ideas in [22] as a point of departure for the work in progress presented here.

The basic operation of pose-control graphs is described in section 2. Section 3 describes how periodic and aperiodic motions are synthesized. Section 4 discusses how the synthesized controllers can be interpolated to produce parameterized motions. Section 5 concludes and gives thoughts on future work.

2.0 Pose-Control Graphs

A *pose-control graph* is a directed graph whose states specify a desired *pose* for a creature to take. The arcs specify conditions upon which transitions between states are taken. A pose consists of a specification of all the internal degrees of freedom of a creature. This is equivalent to all the degrees of freedom, less those necessary to position and orient the creature with respect to the

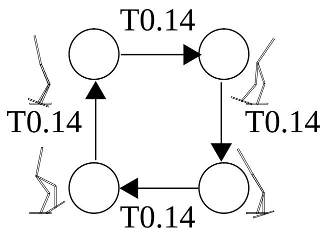


FIGURE 1. A pose-control graph for the walker. The arc labels indicate the duration of the timed state transitions.

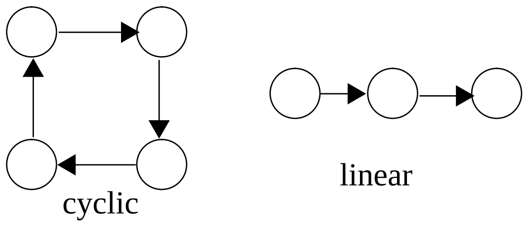


FIGURE 2. Two possible topologies of pose-control graphs.

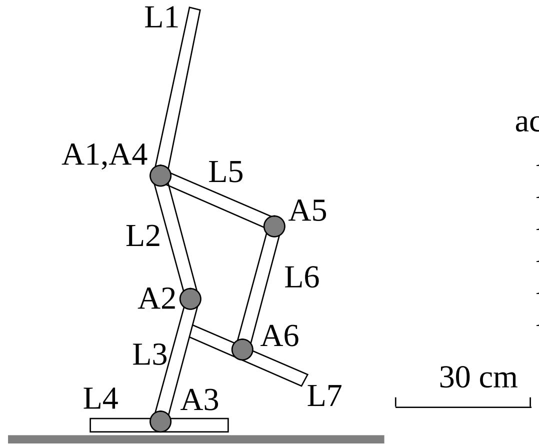
external world. Poses are used as input to a simple low-level control system that is responsible for driving each actuator towards its desired position, typically by exerting torques at rotary joints in our case. Individual proportional-derivative (PD) controllers are used for this purpose. These exert forces or torques equivalent to a spring and damper placed between the desired and actual positions of the given degree of freedom.

A typical pose-control graph is shown in Figure 1. The controller shown produces a walking motion for the seven link *walker* figure, having a total of six actuators. Each state's pose is also shown. Note that the walking figure will never exactly match the specified poses during its motion. Because the poses represent desired internal configurations, we can nevertheless obtain an idea of the expected motion by inspecting the poses alone, just as with keyframes.

Pose control graphs can have several topologies, the simplest of which are shown in Figure 2. Cyclic pose-control graphs are particularly useful for controlling any kind of locomotion because this is usually periodic in nature. Aperiodic motions can be specified using a linear chain of poses. Lastly, more complex pose-control graphs can be constructed through composition, yielding an arbitrary directed graph.

In general, transitions between states can be of several types, although here we shall only deal with timed transitions. This type of transition causes a change of state after a fixed time interval. As such, it is a type of open-loop control. We choose to study time-based transitions because it is important to know what the limitations of open-loop control are before examining the larger space of possibilities that exists when arbitrary sensory feedback is allowed.

A study limited to *cyclic* pose-control graphs with timed transitions was presented in [22], which we now briefly summarize. Several choices in the design of the optimization procedure are analysed. As well, the impact of the design of the physical models upon the motions produced is exam-



Actuators

actuator	min	max	reference
A1	-70	-35	L1 wrt L2
A2	45	80	L2 wrt L3
A3	-40	10	L3 wrt L4
A4	-30	30	L5 wrt L2
A5	45	80	L5 wrt L6
A6	-40	10	L6 wrt L7

FIGURE 3. The walker creature. The reference position representing the zero state for all the degrees of freedom has the figure standing up straight. “wrt” is used to denote “with respect to”.

ined. Lastly, the motions produced by cyclic pose-control graphs are analysed by looking at their bifurcation diagrams. Our present work proceeds onwards from [22] in two different directions. First, we extend the pose-control mechanisms to non-periodic motions. Second, we show that the pose-control graph representation is useful for being able to interpolate and parameterize physically-based motions.

Our experiments are carried out on a class of 2D planar articulated figures, although our initial experiments with 3D creatures have proved to be promising. A creature is specified by modelling its musculo-skeletal structure. In our case, the skeleton is composed of an assembly of rigid links connected with rotary pin joints. The ranges of each joint are specified to restrict the poses of a model to a desired set of allowable configurations. The construction of the walker creature is shown in Figure 3. The muscles are specified by placing actuators of desired strengths at the joints of the model. The strength of an individual actuator is fixed by its spring-and-damper constants, k_p and k_d respectively. The torque exerted is thus given by $T = k_p(q_d - q) - k_d\dot{q}$, where q and $\dot{q}$ are the position and velocity of the degree of freedom associated with the actuator. q_d is the desired position of the joint, which is specified by the current pose. Although we use articulated figures here as models, our method could equally well apply to controlling deformable objects.

3.0 Synthesis of Periodic and Aperiodic Motions

To generate a motion, an animator first creates an actuated, articulated figure, specifies the topology of the pose-control graph and decides on an optimization metric that distinguishes among “better” and “worse” control solutions. Typically this function is some combination of measurable quantities, such as velocity, distance or height of a motion, the stopping distance, the energy consumed, or jerkiness. If the motion is to be periodic, a cyclic graph topology is required. Given a desired period T and a cyclic pose-control graph consisting of n states, initial timed transitions of duration T/n are used. If the motion is aperiodic, a linear pose-control graph using timed transitions is specified a priori.

Supplying an initial timing for a pose-control graph is equivalent to providing some of the information available from a keyframe sketch of the motion. In our case, the number of poses and their

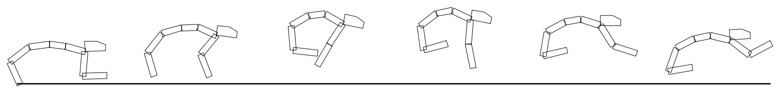


FIGURE 4. A running gait for the 7-link cheetah creature. The motion has been scaled for the purposes of illustration. In the original motion, the creature covers more than twice its body length with each bound.

timing represent important a priori information. In practice, this information serves to greatly reduce the search space (and hence search time) for the synthesis procedure. Similarly, making the pose-control graph cyclic when a periodic motion is desired provides an important constraint on the search space which greatly reduces the search time. As a result, the time and computational power necessary to synthesize solutions can be greatly reduced from those reported in [13] and [20]. It is worthwhile noting that reasonable estimates of the required a priori information could likely be derived automatically from information about the size of the object and the complexity of the motion to be performed.

The synthesis technique must determine the pose for each state such that a desired motion is produced. Our synthesis technique follows the work of [20] in applying a two-phase search, consisting of a global search followed by a local search. Both phases of the search are carried out by using forward simulation trials of the controller in operation. This approach can be described as *generate-and-test* or *modify-and-test*. The technique repeatedly stochastically generates or modifies existing controllers and then evaluates them with respect to the optimization metric by performing a physical simulation. The first phase, making use of generate-and-test, is used to generate a set of candidate controllers that perform best (but probably not well) with respect to the optimization metric. A random candidate controller is generated by assigning randomly-generated poses to the states in the pose-control graph. Typically we choose to generate and test around 100 controllers, retaining the best 10 controllers as being of interest.

The second phase involves using modify-and-test. In our implementation, a randomly-chosen parameter of the controller is perturbed by a fixed delta. The parameter vector to be optimized includes all the degrees of freedom of each pose as well as the transition times between poses. We choose to fix the value of delta to 5 percent of the joint range for pose angles and to 5 percent of the original transition times for the state-transition times. The decision to keep or reject a given controller change is based on the change in resulting performance with respect to the optimization metric. As in [20], both gradient descent and simulated annealing can be used. Gradient descent keeps any changes which result in an improved performance and rejects all others. A more robust optimization scheme is obtained by using simulated annealing[9][20]. In this case, the decision to accept or reject a change for the worse is governed by a stochastic variable and the annealing schedule, which gradually lessens the probability of accepting changes for the worse over time. 200-500 trials are typically carried out during the local optimization phase. As with the work presented in [13], [14], and [20], an optimized physical simulator is necessary to minimize the time necessary to carry out the trials.

Despite the apparent limitations of open-loop control, the motions that can be obtained using cyclic pose-control graphs and the synthesis technique just described can be natural and graceful in appearance, although admittedly this is subjective. Figure 4 shows several frames from a speed-optimized gait for the *cheetah* creature. A surprising result is that this kind of gait can be con-

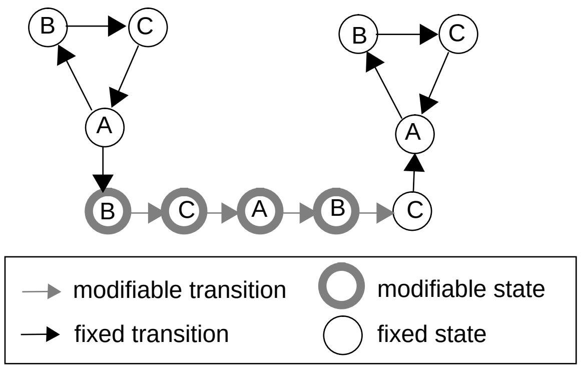


FIGURE 5. Unwinding a pose-control graph. A periodic motion can serve as the basis of an aperiodic motion by unrolling a portion of the cyclic pose-control graph.

trolled in an open-loop manner. It is locally stable with respect to small perturbations, always being drawn back to a terminal limit cycle. The analysis of such local stability has been more extensively studied elsewhere[10]. Other results and analyses of cyclic pose-control graphs are presented in [22]

Many interesting aperiodic motions can be derived from periodic motions. Suppose we wish Luxo the hopping lamp to perform a large leap half-way through a sequence of hops. Let the hopping gait have a cyclic pose-control graph consisting of poses A, B, C, as in the top-left of Figure 5. We now "unwind" two cycles of the motion so that a linear chain of poses exists at the time of the desired leap. After the leap, we reinstate the cyclic hopping motion. We thus have a full pose-control graph as depicted in Figure 5. Four states in the chain are marked as modifiable, as well as the timing of the transitions between these states. These are thus the set of parameters which can be changed by a local optimization (phase two) to allow anticipation, leaping, and recovery for the leap. The leap itself is specified by using the distance travelled over all the hops as an evaluation metric. Because only the parameters associated with one hop are modifiable, this hop is optimized to become a leap. Proper anticipation and recovery is ensured because the entire sequence of hops is simulated during each trial.

The results of the synthesis of a leap are shown in Figure 6. Simulated annealing is used as the optimization technique for this particular motion. The optimization function for this synthesis is the distance travelled over all 7 hops. As well, all changes resulting in a fall are rejected. A fall for Luxo is defined as any part of the body (other than the base) touching the ground. The synthesis of such a leaping motion typically requires on the order of 300 simulation trials, requiring approximately one hour to compute on a modern workstation (~60 SPECfp92).

Surprisingly, the result can also be parameterized. In determining the control for as large a leap as possible, we also have determined the control for any intermediate-size leap. Figure 6d shows the result of such a medium-sized leap, obtained by interpolating between the original and final pose-control graphs of the synthesis process. This kind of parameterization is further discussed in section 4. A useful outcome is that a creature can directly use the results of an 'obstacle detector'

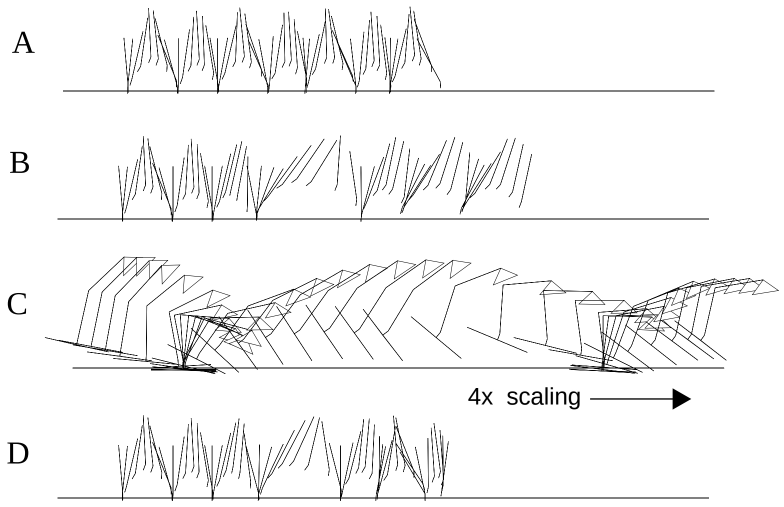


FIGURE 6. Synthesis of a leap. A regular hopping gait for Luxo is shown in (A). For clarity, only the motion of the middle link is shown. (B) shows the result of a leaping motion derived from (A) through optimization. (C) details the anticipation and recovery involved in the leap. (D) shows a smaller leap, obtained by interpolating between the control used for (A) and (C).

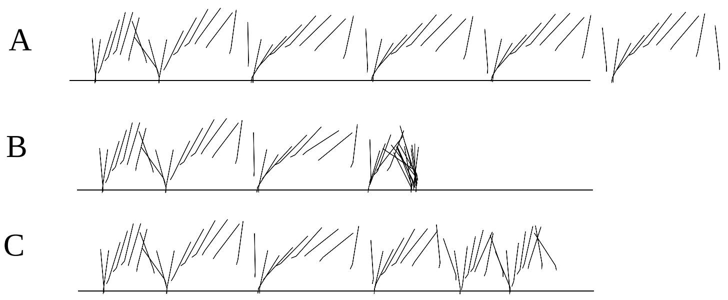


FIGURE 7. Synthesis of a stop. (A) shows a fast hopping gait for Luxo. The middle link only is shown. (B) shows a stopping motion derived from (A) through optimization. (C) shows a slowing-down motion obtained by interpolating between the control used for (A) and (B).

returning the distance-to and size of upcoming obstacles in order to always generate appropriate jumps.

Stopping is an important and largely unexplored aspect of motion. It can be automatically synthesized using our approach. We begin with a fast hopping gait for Luxo. The controller consists of a three-state cyclic pose-control graph. The last two of a total of five cycles are ‘unwound’ and these poses are marked as modifiable. The optimization function to be minimized is in this case the distance travelled without falling over, with a penalty assigned for travelling backwards. This is expressed in $f_{opt} = x_{max} + (x_{max} - x_{end})$, where x_{max} is the furthest distance travelled at any point in the motion, and x_{end} is the final position at the end of the motion.

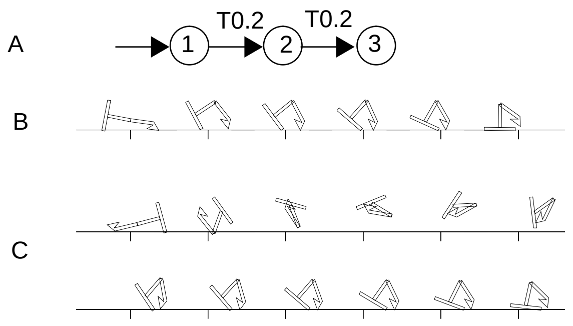


FIGURE 8. Fall recovery for Luxo. The same pose control graph is used for both recoveries (A). The two recovery motions are quite different (B and C), and are shown with different display time steps.

The results of the synthesis process for a stopping motion are shown in Figure 7. The stopping motion is achieved over one hop. Simulated annealing was used to find the parameter modifications necessary to proceed from the original, fast periodic gait to the same gait with a quick stop. It is thus possible to specify a motion such as 'stop as soon as possible' through the use of optimization functions without fixing the final time at which the stop should happen. As with the leaping motion, the original and final pose-control graphs can be interpolated to yield pose-control graphs for intermediate stopping motions. Figure 7c shows a slowing-down motion achieved in this manner.

The two-phase synthesis technique can also be applied directly to the synthesis of non-periodic motions. These are specified using linear pose-control graphs. We assume that the number of poses and an initial estimate of the timing between successive poses is specified in advance. This information could in principle also be automatically synthesized, but we view it as a useful way for an animator to provide a concrete specification of the desired motion. Something that requires more thought for aperiodic motions is the optimization function. In the case of the recovery motions that we shall consider, the goal of a motion is to get the creature from a fallen-down state to one where the creature is back on its 'feet' again. It is necessary in this case to provide an optimization function that will reflect any partial progress made towards the goal.

Figure 8 shows the two possible landing states for Luxo and the synthesized recovery motions. For the Luxo creature, the optimization metric is the deviation of a line passing from the centre of the base to the centre of mass from the horizontal. Both phases of the optimization technique are used, with gradient descent being used to accomplish phase two. The recovery motion from the front requires fewer trials to find than the fanciful backward roll required to recover from its back.

The recovery motion for the walker is shown in Figure 9. There are few possible recovery solutions because of the lack of 'arms' on the walker. The position of the final pose is specified in advance to avoid adding the expense of searching for a suitable stable upright position to the search for the recovery motion itself. The optimization metric to be maximized is the angle of the centre of mass measured in a fashion similar to that used for Luxo. This provides a suitable indicator of partial progress towards the desired recovery motion. Despite being a more complex figure than Luxo, the recovery motions required fewer trials for the walker. This suggests that the required synthesis time is not only a function of model complexity. It is also dependent on the

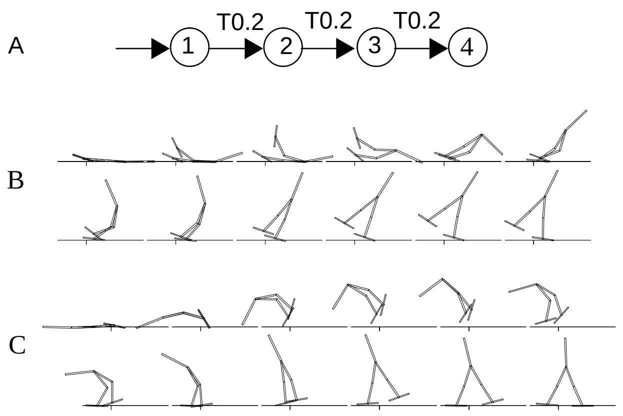


FIGURE 9. Fall recovery for the walker. A 4-state pose control graph is used (A) for both recovery motions (B and C).

“ease” of the problem being solved. We have also had success in using a secondary optimization metric that minimizes the energy used in order to achieve more efficient recovery motions.

The direct synthesis of aperiodic motions using linear pose-control graphs becomes more difficult as the motion involves more states. Whereas periodic motions typically require only three or four poses, lengthy aperiodic motions may easily require more. Each additional parameter in effect adds another dimension to the parameter space to be searched. While the stochastic synthesis techniques are relatively adept at searching in high-dimensional parameter spaces, it is still clearly advantageous to keep the dimension of the search space as low as possible. As a result, complex aperiodic motions are probably best synthesized in several segments. This has been noted earlier in the work of Cohen[4].

4.0 Motion Parameterization

Parameterized motions are an essential way of dealing with the enormous space of all possible motions. It is also a necessary step for being able to build more complex control mechanisms based upon more abstract notions of motion. Thus far we have shown that the animator can influence a motion through the construction of the model and the specification of the optimization metric. In this section we shall show how variations on a gait can be generated without the use of optimization, and how gaits can be interpolated and parameterized using the pose-control graph representation.

There is a complex relationship between the parameters of a pose-control graph and the resulting motions it produces. This relationship is determined through simulation because of the general difficulty in determining it analytically for arbitrary figures. A simulation of the creature with a given pose-control graph yields the resulting motion. A small change in the parameters of the pose-control graph usually leads to a small change in the resulting motion. As a result of this property, it is possible to interpolate between similar motions by interpolating between their controller parameters. We shall define similar motions as any in a set of motions originally derived from the same pose-control graph.

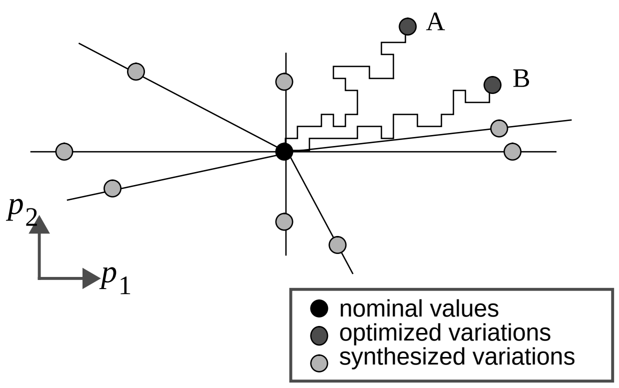


FIGURE 10. Obtaining variations of a motion.

Figure 10 illustrates how a variety of similar gaits can be arrived at. The axes of the figure are defined by two arbitrarily-chosen parameters of a pose-control graph, p_1 and p_2. These may thus represent a degree of freedom in the pose for a state or the timing information for a state transition. The figure shows two different ways in which these parameters can be changed from their nominal values to yield variations of a nominal motion. One method already discussed for changing the values of the parameters is to use optimization techniques. These typically lead to controllers such as those marked A and B in Figure 10. Our current implementation of the optimization techniques only makes changes to one parameter at a time, resulting in the types of parameter trajectories shown.

Another method of automatically generating variations of a motion is to randomly choose a direction in the parameter space and to explore changes in this direction until a similarity criterion is no longer met. This type of exploration results in the synthesized variations of the type shown in Figure 10. In order to use this approach, one needs to specify only a single similarity metric, as opposed to the alternative of specifying a different optimization metric for each variation desired. To date, we have used speed as a similarity criterion. Any gait having a speed within ±30% of the nominal gait is classified as being similar.

In order to be able to effectively generate a random direction in the parameter space, it is necessary to know the sensitivity of the motion with respect to each parameter, in terms of the similarity metric. For this reason, the axes of the parameter space are first explored to determine these sensitivities. The results can be used to normalize the parameter space for the automated synthesis of motion variations.

Once many variations of pose-control graphs have been generated, either through an optimization procedure or the synthesis procedure described above, these variations can then be used as a type of palette by an animator to interactively design new gait variations. The interface we propose for interaction with the motions is based upon similar ideas used for modelling plant-like structures[5], sculptures [6] [19], and abstract images and animations[16][17]. The current instance of the motion being designed and a set of alternative variations are presented at the same time. This type of design-through-selection has previously proved to be an effective interface for exploring other types of high-dimensional spaces[5][16][19]. Figure 11 shows the user interface for this type of gait design. The sliders associated with the nine smaller windows are used to change the control parameters of the pose-control graph being designed towards those of the chosen example. The

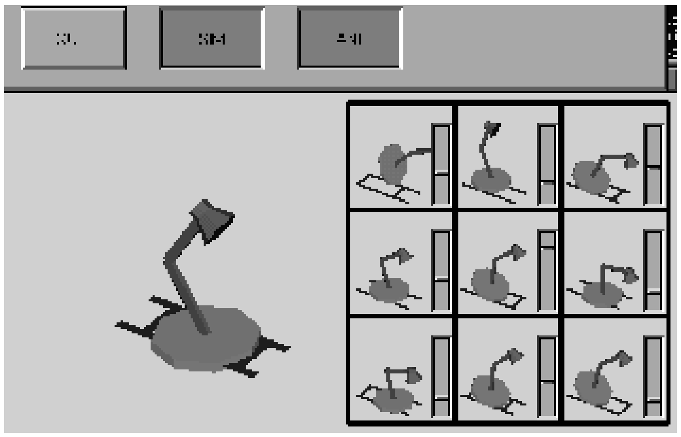


FIGURE 11. The display for interactive gait design. The window on the left displays an ongoing physical simulation of the controller being designed, while the 9 smaller windows play back animations of possible gait variations.

parameters for the pose-control graph being designed are calculated as a linear combination of those of the nominal motion and the variations selected using the sliders.

Using linear combinations of control parameters to create new controllers generally results in predictable motions, but this need not always be the case. If a nominal controller is defined by a parameter vector P_n and a speed-optimal variation of that controller is defined by P_s, then there is no guarantee that a controller defined by $P = kP_n + (1-k)P_s$, $k \in [0, 1]$ will have $v_n \le v \le v_s$, where v_n, v_s, and v are the respective speeds attained by the nominal, optimal, and interpolated controllers. In practice thus far we have found that the motions resulting from interpolated controllers are generally predictable and well-behaved. Convex combinations are generally the most predictable.

A powerful result of the ability to interpolate between gaits in the pose-control graph representation is that it allows for arbitrary parameterizations of motions. For example, as a result of having synthesized a fast gait from a slow one, we also know how to generate a gait of any in-between speed. In a more complex setting, the high dimensional parameter space of the controller can be reduced for control purposes to a low-dimensional parameter space designed by the animator. As an example, a four dimensional parameter space can be designed by choosing four motion variations to serve as the axes of a new four-dimensional space. The nominal motion lies at the origin of this new space. We also foresee parameterized motions being useful in building more complex controllers. Having direct control over more abstract parameters such as speed should simplify the creation of higher levels of control.

Figure 12 shows an example of interpolating between gaits by linearly interpolating the parameters of two pose-control graphs. As described earlier, an interpolated controller is used in a simulation to generate an interpolated motion. The motion which is generated in this fashion thus has the desirable characteristics of an interpolated gait, while at the same time being faithful to the laws of physics.

It is interesting to note that the result of interpolation in the controller parameter space (Figure 12C) yields a different result than kinematic interpolation of individual degrees of freedom between appropriate frames of the two animations (Figure 12D). In general, kinematic inter-

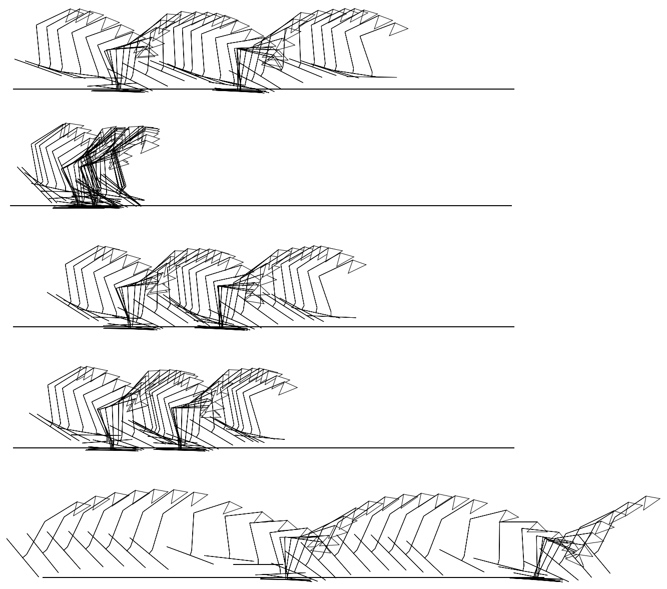


FIGURE 12. Interpolating between gaits. (A) shows the nominal gait for this example. (B) shows a variation of the gait in (A), which tilts the front of the base much further down during a jump. (C) is a gait determined by a controller calculated to lie midway between that of (A) and (B), and then simulating the result. (D) shows the result of using the equivalent kinematic interpolation between (A) and (B). The gait in (C) is further mixed with a fast gait to obtain (E).

polation can cause violations of physical constraints. Pose-control graphs seem to be an effective representation for interpolating between different motions variations while allowing all physical constraints to be properly maintained.

5.0 Conclusions

Pose-control graphs are a flexible control representation for use in creating parameterized periodic and aperiodic motions for physically-based creatures. When used with timed state-transitions, their open-loop control is simple in nature but is capable of producing graceful and complex motions when used with the appropriate synthesis techniques.

While the use of state-machines for control is common, we have shown how to apply optimization techniques in new ways towards the automatic synthesis of desired motions. It has been shown that different kinds of aperiodic motions can be synthesized using periodic motions as a starting point. More importantly, it has been shown that motions can be easily parameterized using the pose-control graph representation. This makes it easier to design higher-level controllers as well as allowing for the interactive design of physically-based motions.

There are several problems with the proposed techniques which have not been addressed. A large class of motions cannot be controlled in the open-loop fashion of pose-control graphs. Motions that are neither statically nor dynamically stable without the use of feedback, such as balancing a pole or riding a bicycle, cannot be controlled using open-loop methods. Such types of motion are in general effectively controlled using continuous state feedback. The synthesis of pose-control graphs through optimization remains a computationally expensive process. Synthesis typically takes on the order of 100-600 trials, each trial chosen to have a duration of 4 or 5 seconds of simulation time. Each such trial typically takes anywhere from 5 - 80 seconds to complete on a modern workstation for planar articulated creatures having 7 or fewer links.[1] It is likely that this could be improved considerably by further refining the optimization process, introducing varying degrees of simulation fidelity at varying costs, and parallelizing the execution of the simulation trials across multiple processors.

How well the synthesis technique presented here scales with the animation of more complex objects remains to be seen. In performing global searches of large parameter spaces, efficient algorithms must take advantage of some underlying structure or constraints of the problem to be solved. We believe that the control representation and algorithms we work with here can be readily extended to take advantage of any structure or constraints that may be used to deal with complex systems. It is worthwhile noting that many articulations in animals are not independently controllable, such as those in the spinal cord. This suggests that the control problem is not as complex as the number of joints in an articulated figure might indicate. Furthermore, it is easy to conceive of simple models as being the initial steps in a coarse-to-fine solution process that terminates with a model of the desired complexity and its controller.

A variety of future work is necessary to further build on the work presented here. We are currently studying how to compose the synthesized motions together in order to obtain more complex autonomous behaviours. The techniques described here also need to be extended to more complex, three-dimensional models and movements. The motions generated through the use of optimization are an equal product of both the optimization technique *and* the physical model. Constructing more complex models of both the figure (skeleton) and its actuators (muscles) is likely necessary to achieve the next degree of realism in physically-based animations. Given that many interesting and useful motions can be synthesized using largely open-loop control, it is interesting to examine in a systematic way exactly what additional control benefits can be obtained by allowing the use of a variety of sensory information. It is not clear what the *best* way of integrating sensory information into a controller should be. Pose-control graphs are structured to use sensory information to make discrete decisions, so this is a first route of investigation.

References

[1] W. W. Armstrong and M. Green, The Dynamics of Articulated Rigid Bodies for Purposes of Animation. Proceedings of Graphics Interface ‘85, 1985, 407-415.

[2] N. I. Badler, B. Barsky, and D. Zeltzer. *Making Them Move.* Morgan Kaufmann Publishers Inc., 1991.

[3] A. Bruderlin and T. W. Calvert. Interactive Animation of Personalized Human Locomotion. In *Proceedings of Graphics Interface ‘93*, 1993, 17-23.

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

[5] R. Dawkins. *The Blind Watchmaker.* Harlow Logman, 1986.

1. Our planar dynamics simulator is currently available through anonymous ftp from: dgp.utoronto.ca in the directory pub/van.

[6] M. Haggerty. Evolution by Esthetics, an interview with W. Latham and S. Todd. IEEE Computer Graphics and Applications, 11, 2 (March 1991), 5-9.

[7] J. K. Hodgins and M. H. Raibert. Biped Gymnastics. The International Journal of Robotics Research, 9, 2 (April 1990), 115-132.

[8] J. K. Hodgins, P. K. Sweeney, and D. G. Lawrence. Generating Natural-looking Motion for Computer Animation. *Proceedings of Graphics Interface '92*, 265-272, May 1992.

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

[10] T. McGeer. Passive Dynamic Walking. *The International Journal of Robotics Research*, 9, 2, 1990, 62-82.

[11] M. McKenna and D. Zeltzer. Dynamic Simulation of Autonomous Legged Locomotion. Proceedings of SIGGRAPH '90. In ACM Computer Graphics, 22, 4 (August 1990), 29-38.

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

[13] J. T. Ngo and J. Marks. Spacetime Constraints Revisitied. Proceedings of SIGGRAPH '93. In *ACM Computer Graphics*, 27 (August 1993).

[14] J. Park, D. Fussell, M. Pandy, and J. C. Browne. Realistic Animation Using Musculotendon Skeletal Dynamics and Suboptimal Control. *Third Eurographics Workshop on Animation and Simulation*, September, 1992.

[15] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *Proceedings of SIGGRAPH '91,* In *ACM Computer Graphics*, 25, 4 (July 1991), 349-358.

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

[17] K. Sims. Primordial Dance. *ACM Siggraph Video Review*, issue 71, segment 26, 1991.

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

[19] S. J. P. Todd and W. Latham. Mutator: A Subjective Human Interface for the Evolution of Computer Sculptures. *IBM United Kingdom Scientific Centre Report 248*, 1991.

[20] M. van de Panne and E. Fiume. Sensor-Actuator Networks. *Proceedings of SIGGRAPH '93,* In *ACM Computer Graphics*, August 1993, 335-342.

[21] M. van de Panne, E. Fiume, and Z. Vranesic. Physically Based Modeling and Control of Turning. *CVGIP: Graphical Models and Image Processing*, 55, 6 (Nov. 1993) , 507-521.

[22] M. van de Panne, R. Kim, and E. Fiume. Virtual Wind-up Toys for Animation, to appear in *Proceedings of Graphics Interface '94*, May, 1994.

[23] J. Wilhelms and B. Barsky. Using Dynamic Analysis for the Animation of Articulated Bodies such as Humans and Robots. *Proceedings of Graphics Interface '8*5, 1985, 97-104.

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

[25] D. Zeltzer. Motor Control Techniques for Figure Animation. *IEEE Computer Graphics and Applications,* Nov. 1992, 53-59.