Dynamic Human Simulation: Towards Agile Animated Characters

Michiel van de Panne
Joe Laszlo
Pedro Huang
Petros Faloutsos

Department of Computer Science
University of Toronto*

Abstract

Dynamic simulation is a potentially useful tool for creating realistic motion for animated characters. However, improved control techniques are required before this approach will bear fruit. We first compare and contrast control for animation with control for robotics. This is followed by an overview of two control methods which were conceived in the context of computer animation, but which also have potential implications for robotic control. Lastly, a forward-looking prospective presents our opinions on what might be needed to make further progress in this challenging area.

1 Introduction

Physically-based models and simulations are behind an increasing number of tools and techniques in computer graphics. Dynamic simulations are particularly well suited to generating passive motions, such as the movement of a character's clothing, the motion of falling and fracturing objects, and the motion of fluids. However, the simulation of active motions remains a formidable challenge because of the control problems involved.

Animation research has nevertheless forged forward, borrowing techniques from areas such as robotics, biomechanics, and control theory, and then integrating them to build motion-control solutions which satisfy the unique needs of computer animation. Figures 1 and 2 illustrate two examples that we shall discuss in detail in this paper. They use appropriately-computed control strategies and dynamic simulation to generate human-like walking behavior and sequential back-flips for a two *acrobot*, respectively.

*{van,jflaszlo,psh,pfal}@dgp.utoronto.ca

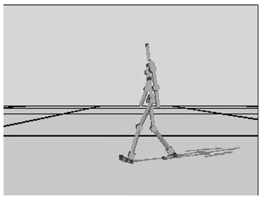

Figure 1: The human model.

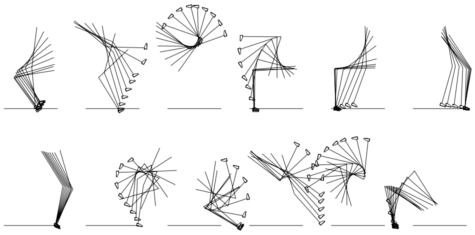

Figure 2: A Flipping Acrobot.

1.1 The Animation Perspective

The task of intelligently controlling the motion of a dynamically-simulated character is in many ways similar (although arguably simpler) to the task of intelligently controlling the motion of a real robot. However, there are some notable differences between the animation problem and the robotics problem. As well, there are many different variations on the animation control problem. The following list elaborates on some of the characteristic axes that define the design space for the motion control problem in animation. Note that the equivalent robotics problem is typically more restrictive when examined in terms of these characteristic axes.

Hard physics vs soft physics.
In creating an animation, a desired motion specification such as "leap 10m forward" might well be impossible. In the context of animation, this need not necessarily be a problem. A clever animation tool can relax the laws of physics just enough to allow the motion to be performed, resulting in a motion that satisfies the motion specification, while still being yielding a 'best fit' to the constraints imposed by physics. Alternatively, animation tools can treat physics as a hard constraint, as implemented by a conventional forward dynamics simulation.

Observed motions vs synthetic motions.
The dynamics involved in creating a motion can be modelled either explicitly, as is the case with typical dynamic simulations, or implicitly, as is the case with good kinematic animation which nevertheless attempt to mimic the appropriate physics. While explicit models are perhaps the most interesting, given that they synthesize motion from first principles, implicit models have the advantage of being able to exploit data captured from real human motions. Motion capture technology has made such data increasingly available, and with the right techniques, captured motions can be generalized to produce novel motions.

Degree of autonomy.
Generating the motion of a foreground character in a feature film is a different problem from generating the motion of a character in a game. The former necessitates detailed control over the motion, while the latter must be capable of interactive autonomous motion. Such different applications are best served by tools uniquely tailored for each, given the different needs. For example, a game character requires models for sensing and planning in order to operate autonomously. These same skills are assumed by an animator when animating a feature-film character.

Visual quality vs. stability
In robotics and control, guarantees of stability are important, while in animation the visual quality is of primary importance.

1.2 Practical Considerations

A number of practical considerations require attention when considering the use of dynamic simulations as a means for creating animation. First, the user interface is necessarily of prime importance. Creating animated motions, whether through traditional animation techniques or using complex dynamic simluations, is a creative process and thus the tools should be predictable in their actions, should provide suitable feedback, and should support the required workflow for creating animations. This is no small challenge, given that these requirements are superimposed on top of already difficult control problems.

The use of physics in computer games presents a further interesting dilemna. Imagine that you are in control of a running character and you suddenly wish to execute a sharp turn to the right. A physically-realistic solution might require waiting until a particular phase in the stride where the sharp turn can be executed. However, from the game-player's perspective, this lends a 'sluggish' feel to the game. The fidelity to physics must be compromised in order to accomodate the game-player's wishes.

A last point of note is that the computer graphics world is not bound by the conventional methodology used in robotics, namely that of treating control and simulation as two separate problems. For the purposes of animation, it may well be more efficient to build a model which encompasses both dynamics and control, thus yielding a black box which accepts high-level motion commands as input and generates appropriate motions as output.

1.3 Explorations

Given the large body of literature which is relevant to the dynamic simulation of human-like characters and robots, it is beyond the scope of this article to provide an overview. Instead, the following two sections present two distinct approaches to control that are drawn from our own work, and which we feel provide useful insights relevant to both robotics and computer animation.

In section 2 we describe the use of *limit cycle control* as a general method for controlling the balance of walking or running gaits. The technique is successfully applied to the control of a 3D dynamically-simulated walking gait, among others. Section 3 presents a technique for planning dynamic motions. The goal here is to provide motions with the anticipatory behavior necessary to execute very dynamic motions across variable terrain. While the results are demonstrated using physical systems having only a few degrees of freedom, they lend potential insight into human motion planning. In section 4, we opine on future directions and methodologies for human simulation and animation. Lastly, conclusions are provided in section 5.

2 Limit Cycle Control

The dynamic simulation of human walking is a surprisingly difficult task, as evidenced by the assumptions and simplifications commonly used in the literature. The limit-cycle control algorithm described here attempts to make relatively few assumptions about the specifics of the gait and the anthropometry. The algorithm is capable of stabilizing walking gaits for dynamically-simulated human models, as well as particular running gaits. A complete description of the method and the results can be found in [3].

The articulated skeleton used for our simulations has 19 degrees of freedom (DOF) and is shown in Figure 1. The dimensions and physical parameters have been chosen to be a realistic reflection of measured human anthropometric data. The character has two DOF ankle joints, three DOF hip joints, and one DOF joints elsewhere. A commercially available dynamics package[2] is used as the core of our simulation and is augmented with procedures for computing the internal and external forces. Internal forces (i.e., torques) are computed using the limit-cycle control algorithm to be described shortly; external ground-reaction forces are computed using a penalty-method contact model. This contact model allows for slippage under the appropriate circumstances.

The point of departure for the control algorithm is a cyclic finite-state machine (FSM) which operates in a deterministic fashion. Each state has associated parameters which specify a set of desired joint angles. The joint torques are then computed using proportional-derivative (PD) controllers, which drive the joints towards their desired configurations. State transitions occur after a fixed time duration, with the exception of two sensor-based transitions. These latter transitions are activated when the left and right feet strike the ground and their function is to keep the FSM synchronized with the movement in progress.

Given appropriate initial conditions and appropriate choices of parameters, the FSM-based controller described thus far is sufficient to make a simulated human take several walking or running steps before falling over due to a lack of balance. The goal of the limit cycle control mechanism is to introduce small control perturbations into the FSM, which will serve to provide the balancing actions.

The operation of the limit-cycle control is illustrated in an abstract fashion in Figure 3. The loop drawn in bold gives the desired periodic motion of the system state. The goal of the controller will be to push the system state back onto this trajectory. Instead of

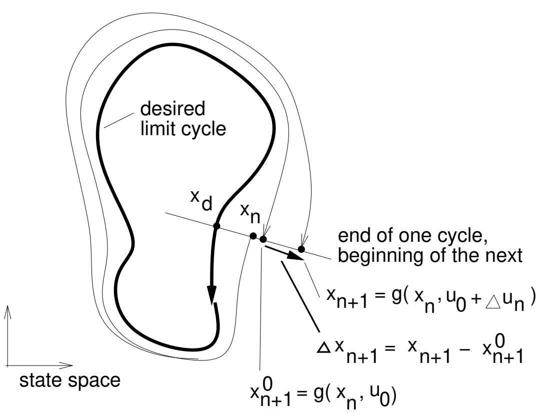

Figure 3: A limit cycle as a discrete dynamical system.

dealing with the continuous system dynamics, however, we can significantly simplify matters by creating a discrete dynamical system. This is done by sampling the motion once per cycle, as illustrated in the figure. The discrete dynamical system can then be expressed as $x_{n+1} = g(x_n, u_n)$, where x_n is the system state at the start of cycle n, x_{n+1} is the system state at the end of cycle n, u_n represents the control inputs applied during cycle n, and the function g is the state transition function. If the original FSM-based controller provides control inputs u_0, then the limit cycle controller must compute the control perturbation Δu_n such that the applied control, $u_n = u_0 + \Delta u_n$, results in the state returning to the desired limit cycle, i.e., $x_{n+1} = x_d$.

In order to further simplify the problem, we introduce the use of a reduced-order model for the system dynamics. First, we note that balance is concerned with avoiding a fall, and thus the basic requirement for x_d is that the simulated human be upright at the end of each step. There are several ways of defining *upright*; one of the simplest is to attach an *up vector* to the coordinate frame of the pelvis. By projecting the up vector onto the ground plane, we obtain two components, one which measures tilt in the lateral plane, i.e., to the left or right, and another which measures tilt in the sagittal plane, i.e., forwards or backwards. We refer to these two observable parameters as the *regulation variables*, as we wish to drive the values of these variables to zero, or some other suitably-chosen target values.

An appropriate choice of *control variables* is required to effect the desired control over the regulation variables. Two control variables are required in order to yield a controllable system. Several choices are possi-

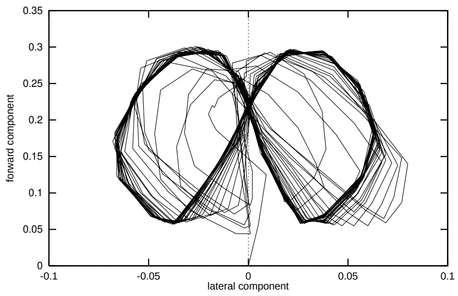

Figure 4: Regulation variable limit cycle.

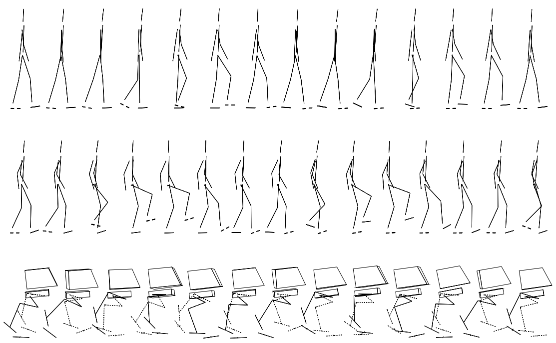

Figure 5: Simulated movements for a human model and robot.

ble; one robust choice we use is to apply an additive perturbation to the target hip angles in one or more of the FSM states. This affects the sagittal and lateral pitch of the trunk segment during the motion and thus can serve as a mechanism for balance control.

Given the definition of the 2D discrete dynamical system, the control algorithm constructs a first order model of the state transition function, g, by evaluating the Jacobian, $\frac{\delta x_{n+1}}{\delta u_n}$, at state x_n. This is done once for each simulated walking step, using a total of four simulations of the same step, each executed with different control parameters. Finite differences are then used to construct the linear model.

An example plot of the controlled system dynamics, as represented by the regulation variables, is shown in Figure 4. The center of the figure-eight pattern represents the target state, x_d. The motion pattern shown is representative of the character swaying to the left and right, as well as to the front and back, as walking steps are taken.

Figure 5 illustrates two variations on a human walk as well as a running gait synthesized for a bird-like robot. Animations of these motions are available on the web[5]. The speed, direction, and style of the walks can be controlled independently of the balancing mechanism. Subjectively speaking, the walks exhibit a rather robotic style, although we intend to explore tuning mechanisms capable of producing more natural walks. Based on the results to date, however, the limit-cycle control algorithm is of potential interest for both animation and robotics applications.

3 Planning Dynamic Motions

The synthesis of regular, periodic gaits such as walking and running is only a starting point for developing the types of dynamic motions which are readily observable in soccer players, tennis players, or cross-country runners. These sports require novel actions that are beyond the scope of a parameterized gait controller and they typically involve interaction with unstructured dynamic environments. In such scenarios, it is difficult to make a distinction between planning and control. Fundamental questions to be asked about such motions include *What action vocabulary is appropriate for planning?* and *How far ahead should one plan?* To this end, we have experimented with a finite-horizon planning algorithm, tested on simplified 'creatures' which are never the less capable of very dynamic motions[1]. One of our creatures is the *acrobot*, a two-link underactuated robot which has been the subject of considerable study in the control literature. The other is *Luxo*, a 'jumping lamp,' drawn from a computer animation context.

We shall explain the motion planner by way of an example. Figure 6 shows the output of the motion planning algorithm for the jumping lamp character. This is a planar articulated figure consisting of three links connected by two single-DOF joints. The problem to be solved is the same for each of the four terrain segments: find the sequence of control actions necessary for Luxo to hop across the terrain in minimal time and without falling. For clarity, Figure 6 only illustrates the motion of the middle link. The final motions, viewable on the web[5], exhibit considerable anticipation of the upcoming terrain, which we shall argue is both realistic and desireable.

The motion planning process is carried out as follows. Luxo is given discrete repertoire of five control actions, which we shall denote as *a, b, c, d,* and *e*. Each of these actions corresponds to a different variation of a forward 'hop.' Qualitatively, these hops vary in their amounts of forward or backward lean. All the hops are implemented using a three-state FSM with fixed, timed state transitions. Each state provides a unique set of desired joint angle values, which are used by

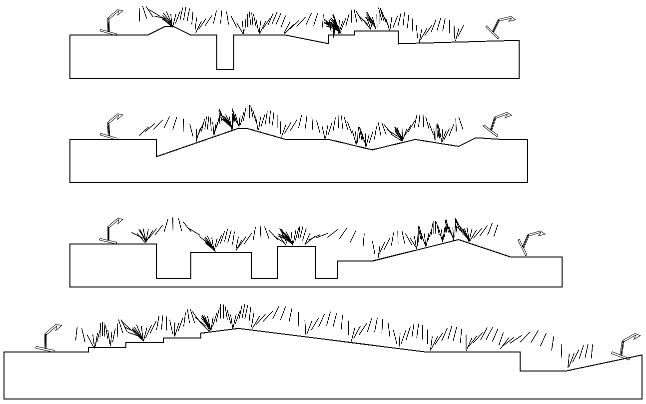

Figure 6: Synthesized control for Luxo on cross-country runs.

local PD controllers on each joint in order to produce the internal torques which drive the motion.

As output, the planning algorithm should produce the sequence of actions which causes Luxo to traverse the terrain in minimal time, e.g., *d,a,b,b,c,a, ...* This is done using a finite-horizon decision tree, which works as follows. A decision tree is constructed to explore the outcome of every possible action sequence of length n, where n represents the planning horizon. For our example, we have chosen $n = 4$, which leads to the simulation of 5^4 possible outcomes. The optimal decision sequence is then chosen based upon the final amount of progress made, i.e., the maximal distance travelled. The planner then commits to the first action of the optimal decision sequence. The entire process is repeated for each subsequent decision. While the decision tree theoretically grows exponentially as a function of the depth, this behavior is not seen in practice because of the early termination of branches due to falls, as well as the application of branch-and-bound techniques.

One of the insights to be gained from the resulting motions is a meaningful evaluation of the required planning horizon for particular classes of motion. The variation of the measured performance, i.e., the time taken to traverse the terrain, can be plotted as a function of the planning horizon used, n. For the examples shown in Figure 6, there are no increasing returns to be had for $n > 4$. An optimal choice of $n = 4$ looks far enough ahead to achieve the best result, while not wastefully planning too far ahead. The current control decision should be influenced by the shape of the terrain up to four jumps ahead, which is a surprisingly long planning horizon.

The same motion planning algorithm also allows for explorations of the capabilities of particular dynamical systems. In particular, we consider the *acrobot*, a simple two-link articulated figure which has a single DOF actuated joint. Novel motions can be synthesized by using the finite-horizon planning process with an objective function which measures progress towards a desired behavior. For example, we have produced control strategies which allow the acrobot to perform sequences of cartwheels, front flips, back flips, and forward balanced hopping. An example of a back-flip sequence is shown in Figure 2. These animations are also available online[5].

It is likely that many human motions (and, by extension, graceful robot motions) will require a planning capability similar to that illustrated here. This is supported in part by the well known presence of anticipatory behavior in human speech (i.e., coarticulations), piano playing, and typing.

4 Forward Perspectives

Together with much other solid work in this area, the motion control techniques discussed in the previous two sections represent a beginning rather than an end. The open-ended nature of the boundaries between the areas of animation, robotics, control, biomechanics, and artificial intelligence ensures that motion control and motion synthesis techniques will continue to evolve in many ways. In this section, we opine on several directions and methodologies for speeding this evolution.

4.1 Wanted: Collaboration

Commencing a research program in the area of dynamic human simulation and motion control is a nontrivial endaevor, requiring the integration of multiple software components. These components include the simulation of articulated-figure dynamics, performing collision detection and response, and rendering graphical output, in addition to obtaining the geometric and physical parameters of the desired human model. This complex infrastructure has served as an impediment to working in this area, as well as making the reproduction of published results problematic.

We believe that a collaborative framework for dynamic human simulation (i.e., shared, publicly-available software) will allow for more focus on the interesting problems, more comparison, more incremental work, and, ultimately, more progress. To this end, we are currently developing such a framework, currently known as DANCE[4].

One of the further long term goals is a formal specification for supporting the composition of multiple motion controllers into a single more complex controller. For example, two users who have independently developed controllers, one for human walking and another for human running, should be able to easily share their controllers in order to construct a more complex, composite controller capable of both actions.

4.2 Machine Learning

There are any number of ways in which machine learning can be used when synthesizing human motion through dynamic simulation. Given that human motion and human behavior is readily observable, it is natural to seek ways of exploiting these observations. The rapid evolution of motion capture equipment has seen the use of motion capture data proliferate in the computer graphics and animation industry. In the short term, captured motions should serve as useful reference motions for tuning the parameters of systems which synthesize motions from first principles. In the longer term, it may be possible to use a large set of captured motions to distill a controller from the data, in effect performing a type of 'controller capture.'

Machine learning can furthermore be applied in cases where the applied control forces are directly observable, as is the case for algorithms such as our dynamic motion planner. By precomputing large families of optimal control solutions and then generalizing from these, machine learning can be used to transform offline optimization techniques into equivalent online behaviors.

4.3 Directing the Control

The demands of computer animation require that dynamic models of humans not only be functional but also be directable, allowing for all types of motion nuances to be achieved. This is an immense challenge, given the innumerable types of motion variations that exist even for common motions such as walking.

5 Conclusions

The dynamic simulation of human motion is a common goal of computer animation, biomechanics, control systems, and robotics. Indeed, these respective fields of research already share many common tools and ideas, although they may differ in their end goals and their means of evaluating success. However, the largely disjoint publication forums for these areas still leaves room for improved collaboration and exchange in order to remove the artifacts that arise from the academic partitioning.

We have described two control techniques which have potential implications for the dynamic simulation of human motion. Limit cycle control illustrates a general method for stabilizing simulated human walks and has the promise of extending to other types of gaits and anthropometry. The finite-horizon motion planner provides insight into the types of motions a dynamical system is capable of, as well as quantifying the amount of anticipation necessary for particular classes of motion.

Lastly, we have briefly speculated on some future directions. A shared framework among research groups would facilitate the easy exchange and incremental improvement of control algorithms for dynamic human simulation. We also (rather timidly) predict a larger role for machine learning in future generations of such control algorithms. Lastly, we emphasize that the interface connecting animators and planning algorithms to controllers is of significant importance.

References

[1] P. S. Huang and M. van de Panne. A search algorithm for planning dynamic motions. *Computer Animation and Simulation '96*, pages 169–182, September 1996.

[2] Symbolic Dynamics Inc. *SD/Fast User's Manual.* 1990.

[3] Joseph F. Laszlo, Michiel van de Panne, and Eugene Fiume. Limit cycle control and its application to the animation of balancing and walking. *Proceedings of SIGGRAPH 96*, pages 155–162, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[4] Victor Ng and Petros Faloutsos. Dance. `http://www.dgp.toronto.edu/dgp/software/dance/dance.html`.

[5] M. van de Panne. Animations web page. `http://www.dgp.utoronto.ca/~van/ani.html`.