Using Natural Vibrations to Guide Control for Locomotion

Rubens F. Nunes
Joaquim B. Cavalcante-Neto
Creto A. Vidal
Federal University of Ceará, Brazil

Paul G. Kry
McGill University, Canada

Victor B. Zordan
University of California, Riverside, USA

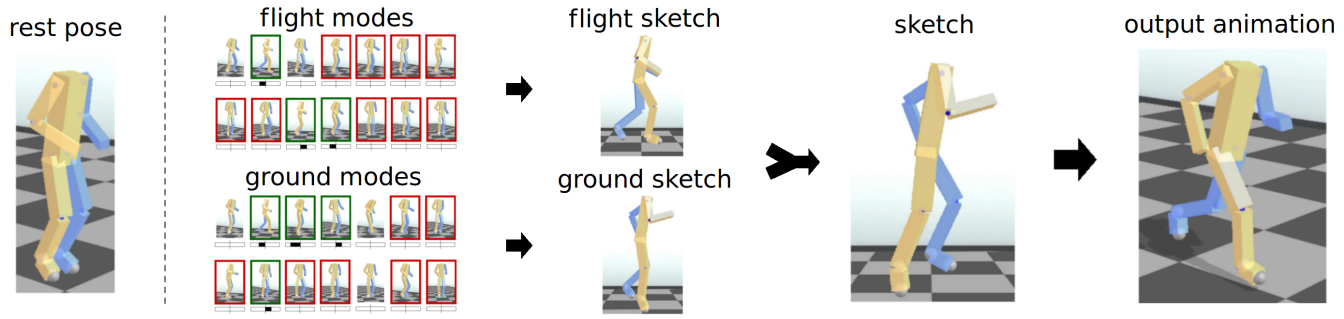

Figure 1: *An overview of our method. We start with a rest pose, and an intuitive selection of specific modes to provide a sketch for the control of a jogging animation. A visual user interface allows an animator to give priority to certain modes (green) as well as ask the system to avoid others (red). An optimization transforms the user sketch of ground and flight phases into a physically based locomotion controller.*

Abstract

Control for physically based characters presents a challenging task because it requires not only the management of the functional aspects that lead to the successful completion of the desired task, but also the resulting movement must be visually appealing and meet the quality requirements of the application. Crafting controllers to generate desirable behaviors is difficult because the specification of the final outcome is indirect and often at odds with the functional control of the task. This paper presents a method which exploits the natural modal vibrations of a physically based character in order to provide a palette of basis coordinations that animators can use to assemble their desired motion. A visual user interface allows an animator to guide the final outcome by selecting and inhibiting the use of specific modes. Then, an optimization routine applies the user-chosen modes in the tuning of parameters for a fixed locomotion control structure. The result is an animation system that is easy for an animator to drive and is able to produce a wide variety of locomotion styles for varying character morphologies.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: motion control, optimization, physically based simulation, modal analysis

1 Introduction

Generating high-quality locomotion for physically based characters is a challenging animation problem because the visual quality of the motion is integrally tied to the function of the controller producing the behavior. In order adjust the components which lead to appearance, an animator must tune the input parameters of the controller without breaking its function. But there is often a non-linear and non-intuitive relationship between the input parameters and the output motion. Thus, while generating a range of motions is possible as in [Sims 1994; Ngo and Marks 1993; Wampler and Popović 2009; de Lasa et al. 2010; Coros et al. 2010], guiding the controller to behave in a specific manner is still a difficult problem.

We propose the use of natural vibration modes as an aid in creating control for locomotion with a specific style. Our hypothesis is that the style of locomotion is derived from naturally occurring oscillations. This claim is supported by other researchers both in robotics [Thompson and Raibert 1990] and in biology [Alexander 1996]. Through a modal decomposition of the dynamics, a palette of oscillations can be generated for the character of interest. Such a decomposition has been used in animation for various applications [Pentland and Williams 1989; James and Pai 2002; Kry et al. 2009; Barbič et al. 2009; Jain and Liu 2011], though none have explored its application for controlling stylized behaviors of physically driven characters.

While the proposed modal basis is merely a reorganization of the character's individual degrees of freedom (with the same size and complexity), it is powerful because it provides a set of natural coordinations that can oscillate together at different frequencies. Kry et al. [2009] show that by focusing on specific low frequency modes, either selected by an animator or automatically, characteristic motions that resemble familiar behaviors arise. In this paper, we exploit this powerful basis to demonstrate its use in the construction of locomotion control for physically based animation.

2 Related work

There has been a vast amount of work on building locomotion controllers for computer animation. The earliest work in this area includes hand-crafted control for legged locomotion [Raibert and Hodgins 1991; Hodgins et al. 1995], evolution of control for virtual creatures [Sims 1994], and optimization for balanced locomotion

[de Panne and Lamouret 1995]. Much of the recent computer animation work focuses on making more robust locomotion control for characters that can perform in various scenarios [Yin et al. 2007; Sok et al. 2007; Da Silva et al. 2008; Muico et al. 2009; Lee et al. 2010; de Lasa et al. 2010; Sok et al. 2010; Coros et al. 2010].

In contrast, our work is more closely related to space-time optimization [Witkin and Kass 1988; Ngo and Marks 1993]. While this previous work uses physical equations of motion as constraints, it is also possible to do such optimization using forward simulation from specified initial conditions. We employ such an approach in this paper to derive control for locomotion with specific characteristics. In contrast to other space-time optimizations, the use of forward simulation allows us to avoid the problem of dealing with non-physical sample trajectories in the search space and frees us from the tricky problem of specifying ground contact constraints within the optimization. Instead, we use a forward simulation which is implicitly constrained to its dynamics and we can employ a standard contact formulation.

Also related to our work is that which finds solutions to constrained motion optimization problems through the aid of reduced search spaces. Safonova et al. [2004] exploit such low-dimensional spaces to create solutions with a specific behavior or style. Their reduced space comes from a principal component analysis of motion capture, while in our work we focus on natural vibrations as a basis. Alternatively, Fang and Pollard [Fang and Pollard 2003] use a set of physics constraints based on aggregate quantities to simplify the optimization problem for efficient synthesis of motion through a set of sketched poses.

Most similar to our own work, Wampler and Popović [2009] also optimize cyclic motion to create a variety of running creatures. One difference is that their work includes morphological parameters in the optimization while our work focuses on generating controllers for a character of fixed morphology. In our work, we propose to limit our search to controllers that exploit the character's morphology in an energy-efficient manner, through its natural modal vibrations. This difference both narrows the search space and grants an animator greater power to guide the outcome of the final motion. Without the modal basis and without the animator input, our framework is very similar to Wampler's. For contrast, we highlight the effects derived from different bases and different degrees of user input in our experimental results. Another example of such cyclic motion optimization can be found in work on controlling the flight of an animated bird along a given path [Wu and Popović 2003] where parametric wing beats are used to structure the search space.

3 Motivation

Research in animation, such as [Raibert and Hodgins 1991], has recognized that the skeletal system stores energy during locomotion through the compression of elastic joints. In recent years, the passive behavior of musculoskeletal structures has been exploited with the goal of improving the quality of character animations [Liu et al. 2005; Kry et al. 2009; Wampler and Popović 2009]. In nature, restorative rebound is a common strategy that appears in natural gaits [Alexander 1988]. This effect is produced by exploiting the musculoskeletal structure to reuse energy stored in opposing muscles, tendons and ligaments. Novacheck [1998] suggests that muscles can be thought to work like springs and dampers, and under this model the restorative energy is derived from the passive return of a muscle "spring" under the influence of its passive stiffness.

Our investigation of modal vibrations is motivated by the claim that beyond the effect of storing energy in individual muscles, the overall shape of skeletal structure plays an important role in how energy is stored. What we see in low energy vibrational modes is the way in which specific structures efficiently use this effect to treat the body holistically as a restorative energy device. Even though this phenomenon merely builds on the same energy "savings" provided by the individual muscles, the savings go farther by coordinating passive dynamic oscillations across the body over time. In this vein, we hypothesize that biological systems exploit natural vibrations in the production of dynamic motion, and that these vibration modes are a useful basis for building locomotion simulations.

4 Control

Our control scheme takes advantage of passive dynamics in combination with the modes through two energy-conserving mechanisms layered over a basic state-machine driven locomotion controller. To produce energy-efficient control, we employ an elastic-restorative model that allows a low-level passive controller to exploit the recovery of energy derived from the elongation of springlike actuators. In addition, we compute a modal basis from the characters' dynamics about a given rest pose, and purposefully actuate our locomotion controller along the coordinations embedded in the modal basis. An added benefit of using a limited number of modes for a specific locomotion controller is that it limits the search space of the control parameters.

We can summarize the process of building the controller (Figure 1) as the following sequence of steps:

- Character specification, to set the hierarchical structure and geometric and physical properties of the character;
- Rest pose definition, to choose the neutral body posture for the character;
- Modal analysis, to calculate the flight modes (Φ^f) and the ground modes (Φ^g) from the chosen rest pose and passive controller gains (Section 4.2);
- Sketch construction, to specify the desired locomotion using *priority* modes and the selection of *excluded* modes, for both flight and ground phases (Section 5);
- Controller optimization, to obtain a cyclic physically based animation controller from the input sketch (Section 6).

4.1 Locomotion controller

We employ a simple locomotion controller which uses a state machine to structure locomotion into flight and ground phases. Within each phase, *active* torques, τ, are applied in short bursts at the beginning of each state, as shown in Figure 2. Torques τ^f are applied for a duration of δ^f seconds in the flight state, and τ^g for a duration of δ^g seconds in the ground state. We choose a square profile to apply the torques for simplicity, but also with inspiration from bang-bang style solutions to optimal control problems. We also experimented with ease-in-ease-out torque profiles to produce smoother cyclic locomotion controllers, but found little benefit.

State transitions are all based on timing, with the period of the entire cycle given as T seconds. The duration of the flight phase is αT and the duration of the ground phase is $(1-\alpha)T$, where $\alpha \in [0,1]$ parameterizes this timing information. For bipedal locomotion, our controller parameterizes one foot only and uses symmetry to provide the timing and activation of the other foot.

4.2 Modal basis and coordination

To achieve our goal of exploiting the character's natural vibrations, we use torques that uphold coordinations derived from a modal de-

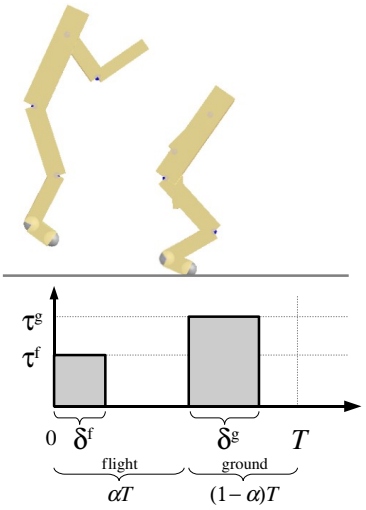

Figure 2: *The locomotion controller state machine has a flight state and a ground state for each foot, with transition times specified at precise times in the period. Torques are applied for a short duration at the beginning of each phase.*

composition and activate only user-selected mode oscillations. In this way, the active torques excite a passive-elastic "muscle" system (Section 4.3) which is then left to produce a restorative response. The most common way to represent the active joint torques is to use a standard full-joint basis with coordinate values representing torques for each specific degree of freedom. We call this representation the *canonical* basis. In contrast, the natural modes of vibration form their own basis, which we employ in our system to compute the active torques. These torques are related to the passive system through the common choice of a neutral rest pose and a set of passive joint stiffness values.

We calculate the modes of a constrained multi-body system with the approach described by Kry et al. [2009]. This analysis provides us with a modal transformation matrix, Φ, in which each column ϕ_i represents one of the natural vibration modes of the character. To match our locomotion phases, we compute Φ^f for the flight modes and Φ^g for the ground modes. The latter uses a bilateral contact constraint positioned at the stance toe to account for the influence of the ground contact in the resulting modes. Thus, at each state, we use the respective modal transformation matrix to get the active torques in joint coordinates from the modal coordinates, ψ, as $\tau^f = \Phi^f\ \psi^f$ and $\tau^g = \Phi^g\ \psi^g$. With the torques represented using this modal basis, we can define our active control simply through the relative torque amplitudes of the desired modes. Intuitively, this allows us to use the modes as a palette of natural coordinations. We use an interface, described below, to build "sketches" of different stylized motions from this palette.

4.3 Passive elastic controller

For restorative response, our controller includes low-level passive torques consisting of a set of simple joint springs with a static neutral pose. This pose is achieved in the absence of gravity and muscle forces and is equivalent to the equilibrium of all passive opposing muscle, tendon and ligament forces.

Along with the user-specified neutral pose, we calculate low-level torques at each joint,

$$\tau_p = k_s(\theta_r - \theta) - k_d(\dot{\theta}), \tag{1}$$

where θ is the current joint angle, θ_r is the rest joint angle, $\dot{\theta}$ is the current joint angular velocity, and k_s and k_d are the stiffness and damping gains. Note, τ_p only depends on the given neutral body posture and passive stiffness, both defined by the user. In all of our examples, we set $k_d = 0.1\ k_s$. At each simulation time step, the active and passive torques are summed and applied to the character.

Through this low-level controller, we account for restorative energy using a simple approach. Namely, any displacement away from the neutral body posture is credited against the cost of the action, while any movement toward the neutral pose is applied to the character without any energy overhead, as if it is simply converting the stored elastic potential energy into kinetic energy.

5 Sketch construction

Our interface provides the animator with a palette consisting of all of the modes, and displays each mode with an animation (see Figure 1). A "sketch" of the desired motion is built by combining modes to form the ground and flight phases. Visual feedback of the resulting sketch is displayed using a simulation of the character without gravity. Building a sketch from modes is straightforward since typically only a few modes are necessary to create a sketch of a desired locomotion style. The accompanying video shows examples of this process.

We call the set of modes selected by the animator the *priority* modes. These modes, which are marked in green in Figure 1, are also assigned nominal amplitudes during the process of building the sketch. Note, a strong benefit of using the modal basis is that each coordinated activation is dynamically independent. When building a sketch, rather than competing, additional modes produce the intended concatenation. Also, in practice, we find that a small number of modes is sufficient to guide a behavior. Beyond the priority modes, we also let the animator choose an explicit set of *excluded* modes (marked in red) that involve motions which are not desired in the final animation. This gives the animator an additional way to direct the result while greatly reducing the search space, and thus improving the speed and tractibility of the optimization task. For clarity in describing our results, we will call all non-excluded modes the *permissible* modes; it is these modes that will have their amplitudes optimized.

In practice, we wish to exclude most high frequency modes, and thus the permissible modes consist of the priority modes plus any remaining low frequency modes that could be potentially recruited during optimization to help with the character's locomotion. The amplitudes chosen for the priority modes in generating the sketch are used as lower bounds for these parameters in the optimization. This ensures that the result of the optimization will have the same characteristics as the sketch. The amplitudes of excluded modes are set explicitly to zero and are ignored by the optimization.

6 Optimization

During natural locomotion, there is always a trade off between performance and energy conservation. We are interested in producing locomotion controllers that are both fast and energy efficient, and we use optimization to find solutions that find a balance to these conflicting goals.

6.1 Optimization formulation

Parameters. The search space for our locomotion controllers includes the following controller parameters:

- q_0 and $\dot{q_0}$, the initial state (position, velocity) of the character, at the start of the locomotion control cycle;
- ψ^f and ψ^g, the torques, in modal coordinates (only the permissible ones), for flight and ground phases;
- δ^f and δ^g, the duration of torques in each phase;
- T, the period of the cycle;
- α, the ratio of the flight phase to total period.

We do not include position or velocity information as parameters of the optimization, except for the initial state. Therefore, unlike other space-time approaches [Liu et al. 2005; Wampler and Popović 2009], we optimize over a search space which only contains physically valid sample trajectories. Also, we choose to optimize symmetric locomotion cycles only. Parameters corresponding to the second foot are set as mirror reflections of the parameters of the first foot. For non-symmetric controllers, we would need to optimize additional parameters in order to account for asymmetries.

Locomotion speed. To ensure that we generate locomotion controllers which are as fast as possible, we optimize for average forward speed,

$$E_v = 1/v_x^2 \,, \tag{2}$$

where v_x is the forward horizontal component of the average center of mass velocity.

Energy expenditure. We quantize energy expenditure with the sum of the squared active torques over the duration of the cycle:

$$E_e = \frac{1}{n} \sum_{t=1}^{n} \sum_{j} {}^t\tau_j^2, \tag{3}$$

where ${}^t\tau_j^2$ is the squared active torque at joint j at time step t. Note that penalizing this sum encourages synchronization with passive response, since the latter is obtained for free and, therefore, is not counted as active energy expenditure.

Constrained optimization. We optimize for both speed and efficiency simultaneously by minimizing the product of our two objective functions over the set of control parameters, which we denote Ω. To produce a cyclic motion, the optimization must be constrained such that the state of the system at the end of the cycle matches the initial state. The optimization problem can thus be stated as follows:

$$\begin{aligned} \min_{\Omega} \;& E_v E_e \\ \text{s.t.} \;& q_0 = q_T \\ & \dot{q_0} = \dot{q_T}. \end{aligned}$$

6.2 Optimization solution

We use a derivative free optimization method which involves sampling the search space. In order to evaluate a sample in the search space, the control parameters are assigned, the forward dynamic simulation is performed, and a fitness value is computed. Our objective function consists of the weighted combination of several terms computed at each time step: the main locomotion objective; a set of *easy* constraints; and a set of hard constraints. The *easy* constraints are used to help convergence and are described in the appendix. The hard constraints simply ensure a cyclic result by matching the final state of the locomotion with the initial state. It is critical that the physical simulation produces a repeatable cycle. We employ the Method of Multipliers (MoM) in combination with Covariance Matrix Adaptation (CMA) to solve this constrained optimization problem.

Covariance Matrix Adaptation. We choose the covariance matrix adaptation evolution strategy [Hansen 2006] as our optimization method because it is well-suited for non-smooth problems with a moderate number of parameters, such as the problem we pose here. CMA does not use or approximate gradients, and does not even presume or require that they exist. This feature has lead to a fair number of uses of CMA in character animation in recent publications [Wampler and Popović 2009; Wang et al. 2009]. In our case, because each evaluation requires running a dynamic simulation, providing a derivative would be problematic (the objective function is not an explicit function of the optimization parameters). Moreover, our objective function is likely to have discontinuities and be subject to many local minima. As such, CMA is a good choice. For brevity, we refer the reader to Hansen [2006] for details on CMA.

The main disadvantage of CMA for us is that it is an entirely unconstrained optimization approach, and therefore it is unable to handle hard constraints such as our cyclic locomotion constraint. In order to overcome this drawback, we employ the Method of Multipliers (MoM). In contrast, Wang et al. [2009] do not handle such hard constraints and Wampler and Popović [2009] handle them by using a hybrid optimization approach combining CMA and space-time constraints.

Method of Multipliers. The method of multipliers [Miele et al. 1972] is a penalty function method that allows a constrained optimization problem to be solved as a sequence of unconstrained ones. Let us consider the problem of minimizing the objective function $f(x)$, subject to the constraint $\varphi(x) = 0$, where $f(x)$ is a scalar, x is an n-vector, and φ is a q-vector with $q < n$. One possible attempt to solve such constrained problem is penalizing how much $\varphi(x)$ is different from zero, by adding some corresponding term $f_\varphi(x)$ into $f(x)$, and solving it as an unconstrained problem. The standard penalty function, for example, is obtained by defining $f_\varphi(x)$ as a term which is quadratic in the constraint: $U(x,k) = f(x) + k\varphi^T(x)\varphi(x)$, where $k > 0$ is the penalty constant.

The main issue with such an approach is the problem of setting the respective weights for the terms, given the trade off that exists between $f_\varphi(x)$ and $f(x)$. While $f_\varphi(x)$ requires a very high influence in order to have the constraint $\varphi(x)$ respected, this might cause main objective $f(x)$ to be neglected. Thus, the idea is to initially prioritize $f(x)$ and gradually update the weights, increasing the influence of $f_\varphi(x)$ in an adequate way. The constrained minimization problem is thus replaced by a sequence of unconstrained minimization problems that in the limit have a minimum point coincident with the solution of the original constrained minimization problem.

In order to circumvent the numerical difficulties associated with the extremely large values of k required at convergence, the method of multipliers uses the augmented penalty function

$$W(x,\lambda,k) = f(x) + \lambda^T\varphi(x) + k\varphi^T(x)\varphi(x), \tag{4}$$

which is obtained by adding to $U(x,k)$ a term which is linear in the constraint $\varphi(x)$. Here, the q-vector λ is an approximation to the Lagrange multiplier. At each optimization, $W(x,\lambda,k)$ is minimized with respect to x for given λ and k. The resulting sample of the current optimization is chosen as the starting sample of the next optimization. Instead of increasing k, it remains the same and the drive toward constraint satisfaction is supplied by automatically updating λ by $\lambda + 2k\varphi(x)$ for the next optimization [Miele et al. 1972]. Thus, convergence can be achieved even with relatively moderate values of k.

In our case, $f(x)$ consists of our main objective and the *easy* constraints, while $\varphi(x)$ is our cyclic constraint:

$$f(x) = w_0 E_v E_e + \sum_s w_s E_s, \tag{5}$$

$$\varphi(x) = \begin{pmatrix} q_T \\ \dot{q_T} \end{pmatrix} - \begin{pmatrix} q_0 \\ \dot{q_0} \end{pmatrix}, \tag{6}$$

where s represents each *easy* constraint, q_T and $\dot{q_T}$ are the final state of the locomotion cycle, and w_0 and w_s are weights. We use $w_0 = 10^{-5}$. Note that any w_s much larger than w_0 (e.g., $w_s > 10^{10}\ w_0$) will work because the easy constraints will eventually be all equal to zero.

To conduct the complete optimization with MoM and CMA, we perform an initialization optimization from any given initial sample with $\lambda = 0$ and $k = 0$, in order to achieve the *easy* constraints and a reasonable initial sample. Then, starting from that sample, we initiate MoM with $\lambda = 0$ and k set to a positive value (we use $k = 0.1$). CMA is used for solving each individual unconstrained optimization and it stops when either a given threshold is met or a maximum number of generations is achieved. The entire algorithm is terminated when, after each individual optimization, either $|\varphi(x)_i| < \varphi_{tol_i}, \forall i$ among the q constraints, or a maximum total number of generations is met. The tolerance φ_{tol} is a q-vector defined by the user.

7 Implementation details

Our approach is not particularly sensitive to the selection of the initial control parameters, Ω. However, for each character, we do assign reasonable initial non-zero values for T, for the δs, and for the height and forward velocity of the root. We initialize $\alpha = 0.5$ and set all the remaining parameters to zero. Symmetric bipeds permit a reduction in the number of control parameters and evaluation time because the parameters for states involving the second foot are reflected versions of the first foot's parameters (i.e., τ, δ, and α). The lag between left and right foot cycles is fixed at half way through the cycle.

Motivated by Wang et al. [2009], we found it practical to employ several soft constraints that are responsible for avoiding ill-behaved regions of the search space. In practice, these constraints are supposed to be *easy* to achieve. Once an easy constraint's term becomes zero, it works like a barrier preventing CMA from entering bad regions. Also, since the value remains zero inside the barriers, it will not influence the other terms, which relieves the need for any tricky weight tuning. Using these constraints, the optimizer is able to find good results even from random initial samples, avoiding the problem of providing a feasible starting sample. Please refer to the Appendix A for more details about the *easy* constraints.

We use the Open Dynamics Engine (www.ode.org) to perform the dynamic simulation and for collision detection. Running times for each CMA optimization is around 1 hour, and the method of multipliers is allowed to take up to 20 optimizations. However, CMA may easily be performed in parallel, reducing significantly the optimization time. Note that this depends on the dimension of the search space that CMA uses to automatically define the number of evaluations per iteration.

8 Results

In this paper, we provide a method for automatically creating locomotion for a given articulated character by optimizing control parameters. To test our method, we produce various controllers with three different fully 3D characters: a biped, a four-armed monster and a kangaroo, with 30, 42 and 36 internal degrees of freedom (DOFs), respectively. The following subsections detail the different approaches we have used to evaluate the capabilities and benefits of our method. Please refer to the accompanying video for the visualization of the examples to better evaluate the results. Filmstrips also appear in Figure 3.

8.1 User-guided sketches

We have created a number of examples that demonstrate how the final output animations are influenced by different locomotion sketches. The video shows the sketch construction of a basic run example for the biped character. It is quite easy to change the look of the sketch, even by changing the amplitude of just a single mode. From the basic run sketch, we can create a variety of different sketches. For instance, the video shows examples of runs that have been modified to include lateral swing, twisting legs, and flapping arms.

In the twisting legs example, we note that the final result involves motion where the whole body twists due to ground contact, which highlights how the optimization finds a control that is physically valid while respecting the sketch. In the lateral swing example, we also suppress ground phase arm swing by excluding a specific mode, demonstrating how the final motion is directable both in the sense of the motion we want, as well as motions we do not want.

We find that even when our optimization is given a random motion sketch, it is still able to successfully find a result which follows the style of the sketch. The key observation here is that our optimization step is not just finding an arbitrary optimal solution, but is respecting the sketch while producing a physically valid and functional motion in spite of how unusual the sketch might be. We show examples in the accompanying video.

8.2 Changing the rest pose

Editing the rest pose of a character is an easy way to change the resulting optimized motion. The video shows an example where we use a "squat" rest pose and create a sketch similar to the basic run to produce a squatting locomotion gait. We also created a one-legged hopping gait. For this example, a one-leg rest pose is crafted, and a modified single-leg version of our biped locomotion controller is defined in order to specify that only one leg contacts the ground during each cycle.

8.3 Changing the character

Our method is general and applicable in a straightforward and easy way for different characters with different morphologies. Beyond a normal biped, we show examples for a monster with four arms and a kangaroo (Figure 3). For the monster, we show that we can easily change the coordination style of the arm swings by including different modes in the sketch. Note that the motion is dynamically adapted in order to obtain the new swing of arms, in an automatic way. For the kangaroo character, which presents a different morphology, we produce a hopping gait using the single-leg controller described.

8.4 Comparison with other approaches

Table 1 shows how many modes we use in each phase of the controller for the different examples described. Note the search space is much smaller when optimizing only permissible modes, instead of optimizing by using the canonical (full-joint) basis representation.

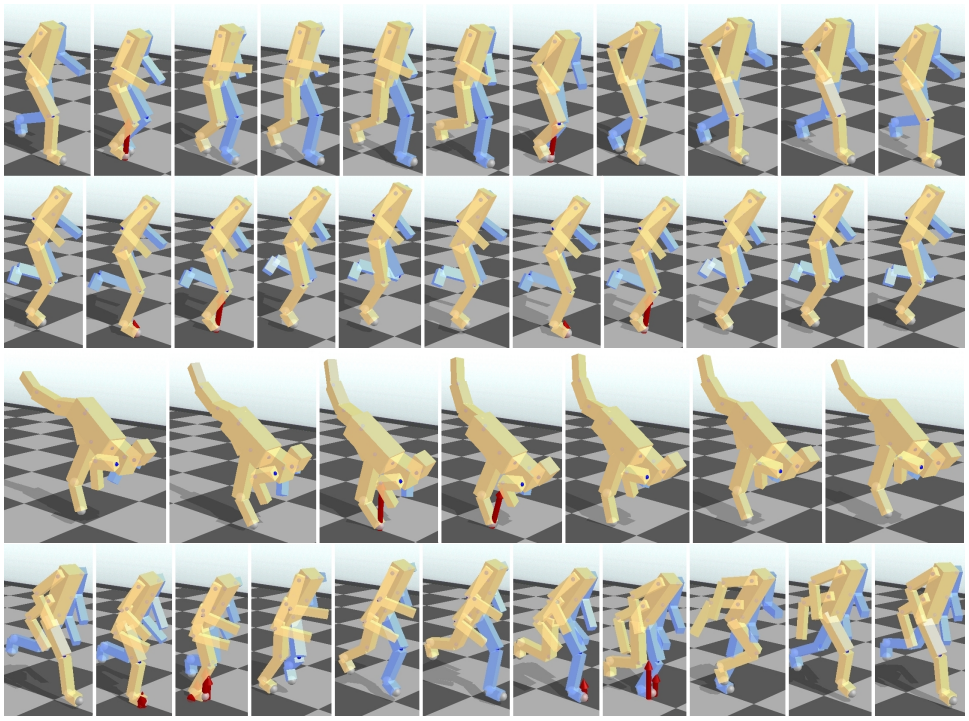

Figure 3: *Filmstrips from animation examples.*

Table 1: *Number of selected modes for each set of modes at flight and ground states (f/g).*

Example	Priority (f/g)	Permissible (f/g)	Canonical (f/g)
runner	3/4	5/7	30/30
hopper	2/5	6/8	30/30
monster	3/4	5/7	42/42
kangaroo	1/3	4/6	36/36
"naive" runner	3/4	12/12	30/30

Modal vs canonical basis. In order to justify the use of the modal coordinates to represent the active torques, we compare the use of the canonical basis in optimizing our basic run example. When using the canonical basis we lose control over the desired oscillations that will appear at final motion. While the optimized result is valid, it does not necessarily match a recognizable or desired style.

Permissible modes - user-guided vs "naive" selection. As mentioned, the exclusion of specific modes allows the animator to prevent the use of undesired motions. To evaluate this feature, we re-examine our basic run example, but this time considering all of the first m modes at each state as a naive set of permissible modes. Comparing the optimized results, we observe that the naive selection of the permissible modes does not restrict the resulting animation in the desired way. Thus, we conclude that leaving the choice of the permissible modes as a task for the animator is important for guiding the desired result.

On the other hand, if we optimize using only the priority modes as the permissible modes, achieving the desired cyclic controller becomes difficult for the optimizer. This is because the available controller space is too narrow for the optimizer to find a valid result. However, as shown in Table 1, we find that our examples typically only need a few extra modes in addition to the priority modes.

9 Discussion and conclusion

In contrast to many other techniques for controller production, our method does not use any reference motion data (i.e., keyframes or captured motions) to generate the resulting control. For each character and desired locomotion, the final outcome is derived from only a single rest pose and the input mode-based sketch.

While activating in the directions of the low frequency modal coordinations realizes energy efficient control through restorative energy across passive springs, we also observe reduced interference between the individual coordinations due to the orthogonal properties of the modes. This phenomenon is key in controlling the visual appearance of the outcome of the controller because the modal basis provides a palette of choices that can be selected with weighted priorities by a user. Assuming that a control can be found, these modes appear with little mutual interference in the final motion, since they are dynamically independent from one another. Thus, by choosing and specifying modes, an animator can yield a high-level control over the coordinations present in the output motion.

One limitation of this work is in the simplicity of the locomotion controller. Recall that the ground phase thrust occurs at a specific time in the cycle. While we can think of these ground phase torques as pumping the system at some fixed delay after the time of contact (and may prefer a ground contact plus delay trigger as a parameterization to use the controller in an interactive simulation), recall that we are primarily concerned with finding controller parameters that are tuned by optimization. Thus, using simply the timing of the ground thrust as a parameter alleviates the need to identify the time of contact in the optimization, and likewise any sensitivity to variations in the time of contact. All of this is also the case for the timing of the flight phase torques as the cycle starts from a given pose, q_0, with initial velocity, $\dot{q}_0$, which needs not be aligned with the time at which the foot breaks contact with the ground. In our case, unlike space-time constraints approaches, contacts spontaneously

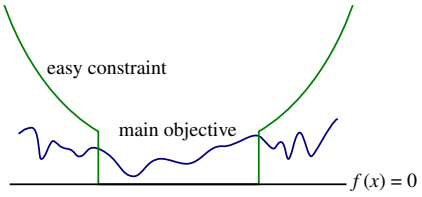

Figure 4: *Easy constraint represented by a bilateral thresholded quadratic function. Easy constraints (green) are added to the main objective (blue) to help find the global minimum.*

arise from the forward dynamic simulation and therefore need not be explicitly addressed in the optimization.

Guiding the control of a physically derived motion will remain a challenge because it will always be at odds with the appearance of the motion itself. However, in this paper we make headway on this problem by exploiting natural coordinations that appear in the modal basis of a physical system. Our hypothesis is that natural vibration modes contribute to energy efficient movement and indeed we see that especially low frequency modes share similarities with behaviors such as running and hopping (as well as other oscillatory movements). We further posit that the modal basis, especially low frequency modes, would therefore represent a good basis for describing and building desired motions. With this set of ideas in mind, we present a method for providing an intuitive visual mechanism to animators in order to allow them to choose and guide the development of physically driven character locomotion. To realize this effort, we employ an optimization routine which employs two methods (CMA and MoM) for honing in on a controller that can produce locomotion with a desired visual appearance.

Acknowledgments

This work was supported by CNPq (process 200956/2008-6), CAPES, NSERC, and GRAND NCE Project MOTION.

A *Easy* constraints

Intuitively, *easy* constraints represent steep penalty slopes placed over the ill-behaved regions in order to improve the behavior of the objective function, helping the optimizer find the desired regions (see Figure 4). Following the work of Wang et al. [2009], we represent easy constraints using thresholded quadratic functions. We define an unidirectional upper bound constraint function

$$Q_u(d,\epsilon) = \begin{cases} (\epsilon - d)^2 + h, & \text{if } d < \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where the height of the "barrier" is set by h and the weight associated with each *easy* constraint. We find that $h = 1$ works well for all of our constraints because the weights are all quite large. We use $Q_u(-d,-\epsilon)$ to define a unilateral lower bound constraint, with which we can write a bilateral function $Q_b(d,\epsilon) = Q_u(-d,-\epsilon) + Q_u(d,-\epsilon)$, which is analogous to checking if $|d| > \epsilon$. In the remainder of this section, we describe each of the different *easy* constraints that we employ.

Ensure COM velocity. We penalize small minimum and average forward speed as well as large average vertical and lateral speeds. Because the objective is for the character to run in the positive x direction, these functions are defined as

$$E_1 = \sum_{t=1}^{n} Q_u({}^t\dot{c}_x, 0.5), \tag{8}$$

$$E_2 = Q_u(v_x, 2.0) + Q_b(v_y, 0.25) + Q_b(v_z, 0.25), \tag{9}$$

where ${}^t\dot{c}$ is the velocity of the center of mass at time step t, and v is the average center of mass velocity.

Maximum joint velocities. For efficient locomotion, we find that very large joint velocities are undesirable. With ${}^t\dot{\theta}_j$ giving the angular velocity of joint j at time step t, we write this constraint as

$$E_3 = \sum_{t=1}^{n} \sum_{j} Q_b({}^t\dot{\theta}_j, 20.0). \tag{10}$$

Maximum energy. A maximum active energy value is set in order to avoid very energetic regions of the search space, which are likely to be less well-behaved:

$$E_4 = Q_u(-E_e, -500000.0). \tag{11}$$

Ensure priority modes. The user's sketch must be respected and not dominated by other permissible modes, thus, we constrain the search to be within the subspaces dominated by the priority modes. For each state s, to ensure that the sum of the unsigned amplitudes of the priority modes is greater than the sum of the unsigned amplitudes of the other modes, we define the function

$$E_5 = \sum_{s} Q_u \left(\sum_{p} |\psi_p^s| - \sum_{o} |\psi_o^s|, 0.0 \right), \tag{12}$$

where ψ_p^s are the amplitudes of all priority modes at state s, and ψ_o^s are the amplitudes of the other modes at state s.

Ensure upright posture. We penalize the character if it is not upright. With the height of the torso, ${}^tp_y$, and the height of the center of mass at time step t, ${}^tc_y$, we write this constraint as

$$E_6 = \sum_{t=1}^{n} Q_u({}^tp_y - {}^tc_y, 0.0). \tag{13}$$

Ensure stance. The active foot must be in contact with the ground during some portion of stance. We guarantee that a minimum distance between foot and ground, ${}^td_y$, is met during the n' time steps following the start of ground thrust at t' using

$$E_7 = \sum_{t=t'}^{t'+n'} Q_u(-{}^td_y, -0.001). \tag{14}$$

Minimum toe height. The character must leave the ground during the cycle. We measure this by computing the air-travel distance of the toe, δ_y:

$$\delta_y = \sum_{t=1}^{n} {}^tk_y({}^td_y - {}^{t-1}d_y), \tag{15}$$

where ${}^tk_y = 1$ for the active toe if specific conditions are met. Namely, $\dot{c}_y$ must have already gone negative since the toe became active; ${}^t\dot{c}_y$ must be positive; ${}^t\dot{c}_y - {}^t\dot{d}_y > -0.1$; ${}^th_y > {}^td_y$; and ${}^td_y > {}^{t-1}d_y$. Otherwise, ${}^tk_y = 0$. Here, ${}^th_y$ is the minimum distance between the heel and the ground and ${}^t\dot{d}_y$ is the active toe's

vertical speed. Finally, we penalize this term, δ_y, if it is below a certain value using

$$E_8 = Q_u(\delta_y, 0.02). \tag{16}$$

Avoid toe stub. For biped locomotion, we ensure that the swing foot does not stub the floor by keeping the hip to toe distance of the swing leg shorter than that of the stance leg during the ground contact:

$$E_9 = \sum_{t=t_g}^{t_f} Q_u(l_g - l_f, 0.0), \tag{17}$$

where t_g is the time the stance foot hits the ground and t_f is take-off, and l_g and l_f are the hip to toe distance of the stance and swing leg, respectively.

References

ALEXANDER, R. M. 1988. *Elastic Mechanisms in Animal Movement*. Cambridge University Press.

ALEXANDER, R. M. 1996. *Optima for Animals*. Princeton University Press.

BARBIČ, J., DA SILVA, M., AND POPOVIĆ, J. 2009. Deformable object animation using reduced optimal control. *ACM Trans. Graph. 28* (July), 53:1–53:9.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2010. Generalized biped walking control. *ACM Trans. Graph. 29* (July), 130:1–130:9.

DA SILVA, M., ABE, Y., AND POPOVIC, J. 2008. Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum 27*, 2.

DE LASA, M., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-based locomotion controllers. *ACM Trans. Graph. 29* (July), 131:1–131:10.

DE PANNE, M. V., AND LAMOURET, A. 1995. Guided optimization for balanced locomotion. In *Proceedings of Eurographics Workshop on Computer Animation and Simulation*, 165–177.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Trans. Graph. 22* (July), 417–426.

HANSEN, N. 2006. The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms*, Springer, 75–102.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '95, 71–78.

JAIN, S., AND LIU, C. K. 2011. Modal-space control for articulated characters. *ACM Trans. Graph. 30*, 5, 1–12.

JAMES, D. L., AND PAI, D. K. 2002. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph. 21* (July), 582–585.

KRY, P. G., REVERET, L., FAURE, F., AND CANI, M. P. 2009. Modal locomotion: Animating virtual characters with natural vibrations. *Computer Graphics Forum 28*, 2, 289–298.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Trans. Graph. 29* (July), 129:1–129:8.

LIU, C. K., HERTZMANN, A., AND POPOVIĆ, Z. 2005. Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans. Graph. 24* (July), 1071–1081.

MIELE, A., MOSELEY, P. E., LEVY, A. V., AND COGGINS, G. M. 1972. On the method of multipliers for mathematical programming problems. *Journal of Optimization Theory and Applications 10*, 1, 1–33.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graph. 28* (July), 81:1–81:9.

NGO, J. T., AND MARKS, J. 1993. Spacetime constraints revisited. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '93, 343–350.

NOVACHECK, T. F. 1998. The biomechanics of running. *Gait and Posture 7*, 1, 77–95.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: modal dynamics for graphics and animation. *SIGGRAPH Comput. Graph. 23* (July), 207–214.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. *SIGGRAPH Comput. Graph. 25* (July), 349–358.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. Graph. 23* (August), 514–521.

SIMS, K. 1994. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '94, 15–22.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graph. 26* (July).

SOK, K. W., YAMANE, K., LEE, J., AND HODGINS, J. 2010. Editing dynamic human motions via momentum and force. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '10, 11–20.

THOMPSON, C., AND RAIBERT, M. 1990. Passive dynamic running. In *Experimental Robotics I*, V. Hayward and O. Khatib, Eds., vol. 139 of *Lecture Notes in Control and Information Sciences*. Springer Berlin / Heidelberg, 74–83. 10.1007/BFb0042513.

WAMPLER, K., AND POPOVIĆ, Z. 2009. Optimal gait and form for animal locomotion. *ACM Trans. Graph. 28* (July), 60:1–60:8.

WANG, J. M., FLEET, D. J., AND HERTZMANN, A. 2009. Optimizing walking controllers. *ACM Trans. Graph. 28* (December), 168:1–168:8.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. *SIGGRAPH Comput. Graph. 22* (June), 159–168.

WU, J.-C., AND POPOVIĆ, Z. 2003. Realistic modeling of bird flight animations. *ACM Trans. Graph. 22* (July), 888–895.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. Simbicon: simple biped locomotion control. *ACM Trans. Graph. 26* (July).