解 説

Efficient Computation of Optimal, Physically Valid Motion

Anthony C. Fang[*1] *and* Nancy S. Pollard[*2]
[*1]Department of Computer Science, National University of Singapore [*2]Robotics Institute, Carnegie Mellon University

1. Introduction

One appealing vision is that a user should be able to design a motion trajectory by setting a small number of keyframes and constraints—and that the resulting motion should remain optimal in some way. Variations of this problem have appeared in computer graphics, biomechanics, and robotics (e.g., [9] [6] [8] [7] [4]) Despite great interest and research progress, however, optimization is still viewed as impractical when the number of degrees of freedom is high; the reputation is that optimization will be slow, cumbersome, and difficult to guide toward a desired solution.

To obtain motion for high degree of freedom characters using optimization, researchers in computer graphics, for example, have explored techniques ranging from extreme model simplification (e.g., [8]) to enforcing stereotypical momentum patterns [5] to discarding physical constraints entirely and relying on the animator to give the character a sense of weight and balance (e.g., [3]).

We have found that physically plausible motion can be created for high degree of freedom characters by choosing to use only constraints and objective functions with derivatives that can be computed in time linear in the number of degrees of freedom of the character. Constraints that fall within this set include many of those used to enforce correct physics. For example, the swinging character in Figure 1 can apply very little torque about the bar axis, ground contact forces should fall within the friction cone at the feet, and linear and angular momentum should be conserved during the dismount. As the basis of our algorithm, we present a new formulation of the equations of motion that allows first derivatives of constraints such as these to be computed in $O(D)$ time, where D is the number of degrees of freedom (DOF) of the character, vs. the $O(D^2)$ time expected from direct analytical differentiation, numerical differentiation, or automatic differentiation. We add to the existing body of research this $O(D)$ algorithm for computing first derivatives of a broad range of physics constraints for improved performance in a optimization context (also see [1]).

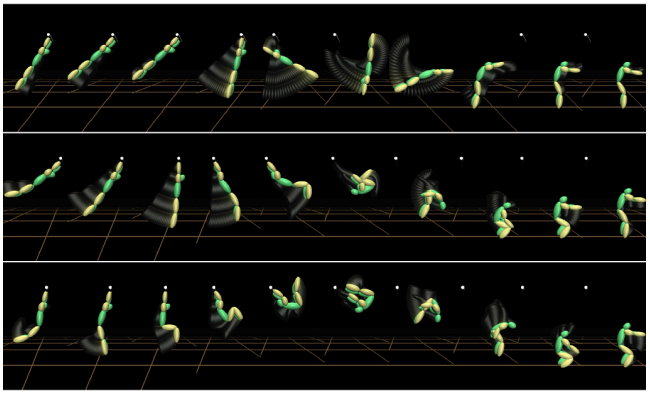


Fig. 1 A single-flip dismount from a high-bar. (Top) Initial guess based on kinematic interpolation of full body motion. (Middle) Flight duration is 0.6 seconds; Flip posture is tight and maximum height attained is below high-bar. (Bottom) Flight duration increased to 0.8 seconds. Flip posture is relaxed, and maximum height attained exceeds high-bar.

The theoretical speedup from $O(D^2)$ to $O(D)$ is possible because the individual torques at the character's joints are not computed—physics constraints are formulated based on the aggregate force and torque applied by the character to the environment. Because individual joint torques are not available, our objective function should not depend explicitly on those values. Our results suggest, however, that physics constraints and a kinematic measure of smooth motion are sufficient to capture dynamic effects such as squash-and-stretch and

原稿受付
キーワード：animation, physically based animation
[*1]Block S14 #06-08 3 Science Drive2, Singapore, 117543
[*2]5000 Forbes Ave., Pittsburgh, PA, 15213, USA

tucking for faster rotation, as shown in Figure 1.

2. Efficient Physics Constraints

We state the optimization problem solved at each stage in the following form:

$$\min_x h(B(t)x) \text{ subject to } c(t_i) = 0, i = 1..m, t_i \in [t_s, t_f]$$

where h is the optimization function; $B(t)$ is a set of basis functions; x are the coefficients, the free parameters of the optimization; and $c(t_i)$ are the constraints. We use B-splines as basis functions and follow the standard approach of enforcing constraints at a fixed number of keyframes.

Constraints that enforce physical validity can be formulated as linear equality or inequality constraints on aggregate force. The aggregate force is a representation of all external forces and torques (excluding gravity) that would have to be applied to the character root to explain the character's motion. For example, when the character is swinging on a high bar or monkey bars, the amount of torque that can be applied about the bar axis is constrained. Let aggregate force f_0 be represented as

$$\mathrm{f}_0 = \begin{bmatrix} \underline{\mathrm{f}}_0^a \\ \underline{\mathrm{f}}_0^b \end{bmatrix} \tag{1}$$

where $\underline{\mathrm{f}}_0^a$ is linear force and $\underline{\mathrm{f}}_0^b$ is torque about the world origin. Aggregate force is translated to a constraint point $\underline{\mathrm{c}}$ as follows:

$$\mathrm{f}_c = \begin{bmatrix} \underline{\mathrm{f}}_c^a \\ \underline{\mathrm{f}}_c^b \end{bmatrix} = \begin{bmatrix} \underline{\mathrm{f}}_0^a \\ \underline{\mathrm{f}}_0^b - \underline{\mathrm{c}}^0 \times \underline{\mathrm{f}}_0^a \end{bmatrix} \tag{2}$$

where $\underline{\mathrm{c}}^0$ is the world vector from the origin to $\underline{\mathrm{c}}$.

The bar contact constraint can then be expressed as

$$-\tau_{max} < \underline{\mathrm{s}}_{bar} \cdot \underline{\mathrm{f}}_c^b < \tau_{max} \tag{3}$$

where τ_{max} is the scalar torque limit, $\underline{\mathrm{s}}_{bar}$ is the bar axis, and $\underline{\mathrm{s}}_{bar} \cdot \underline{\mathrm{f}}_c^b$ is a projection operation that results in torque about the bar axis.

When constraints are formulated as functions of aggregate force, they can be evaluated efficiently using any efficient inverse dynamics algorithm. However, at each stage of the optimization, derivatives of constraints and objective functions may also be required. For example, the sequential quadratic programming algorithm used in [9] makes use of first derivatives of the constraints and both first and second derivatives of the objective function. Given free parameters x, the B-spline coefficients defining the character's trajectory, first derivatives of constraints are simple functions of first derivatives of aggregate force, which is expressed as follows:

$$\frac{\partial \mathrm{f}_0}{\partial x} = \boxed{\frac{\partial \mathrm{f}_0}{\partial q}} \frac{\partial q}{\partial x} + \frac{\partial \mathrm{f}_0}{\partial \dot{q}} \frac{\partial \dot{q}}{\partial x} + \frac{\partial \mathrm{f}_0}{\partial \ddot{q}} \frac{\partial \ddot{q}}{\partial x} \tag{4}$$

At any time t, character position q, velocity $\dot{q}$, and acceleration $\ddot{q}$ are known, and terms $\partial q/\partial x$, $\partial \dot{q}/\partial x$, and $\partial \ddot{q}/\partial x$ are available trivially from the definition of B-splines. The term $\partial \mathrm{f}_0/\partial q$, which we will refer to as the *force Jacobian*, is the most difficult term in this expression, and the trick is to compute this term in time linear in the degrees of freedom of the character.

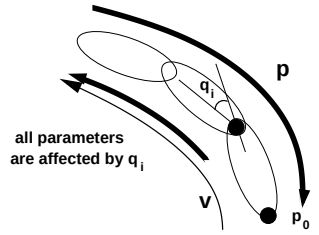


Fig. 2 The effect of parameter q_i is propagated up the tree with velocities v and back down the tree with momentum terms p. Computing $\partial \mathrm{p}_0/\partial q_i$ requires $O(D)$ time and results in an $O(D^2)$ algorithm for computing the momentum Jacobian.

2.1 Linear Time Momentum Jacobian

Efficiently computing $\partial \mathrm{f}_0/\partial q$, the force Jacobian, requires efficiently computing $\partial \mathrm{p}_0/\partial q$, the momentum Jacobian, because aggregate force f_0 is the time derivative of aggregate momentum p_0.

The usual way to compute aggregate momentum is to formulate the following recursion:

$$\mathrm{v}_i = \mathrm{X}_{i-1}^i \mathrm{v}_{i-1} + \mathrm{s}_i' \dot{q}_i \tag{5}$$

$$\mathrm{p}_i = \mathrm{X}_{i+1}^i \mathrm{p}_{i+1} + \mathrm{I}_i' \mathrm{v}_i \tag{6}$$

where p_0 is the desired result. Note that we use spatial vector notation as in [2], and so both linear and angular momentum terms are represented in Equations 5 and 6.

Velocities v_i are propagated from base to leaf, and momentum p_i is propagated from leaf to base. Figure 2 shows this process. Parameter q_i appears in the coordinate transforms X_{i+1}^i and X_i^{i+1}, and so every v_j for $j > i$ depends on q_i, and every p_j for $j \geq 0$ depends on q_i. Unrolling the recursion to collect terms for $\partial \mathrm{p}_0/\partial q_i$ requires $O(D)$ time. There are D terms q_i,

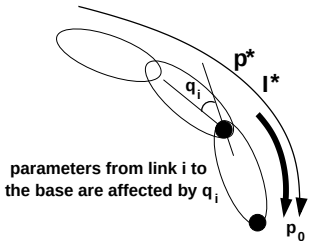


Fig. 3 The effect of rewriting the recursion is to limit the effect of q_i to parameters collected at joints between i and 0. Terms required for the momentum Jacobian are accumulated in a single pass from leaf to base, and the momentum Jacobian can be computed in linear time.

and this approach will lead to an $O(D^2)$ computation for the momentum Jacobian. There is no clever way to simplify the calculation by aggregating terms when it is presented in this form.

We observe that rewriting the recursion solves this dilemma:

$$\mathrm{I}_i^* = \mathrm{X}_{i+1}^i \mathrm{I}_{i+1}^* \mathrm{X}_i^{i+1} + \mathrm{I}_i' \quad (7)$$

$$\mathrm{p}_i^* = \mathrm{X}_{i+1}^i \mathrm{p}_{i+1}^* + \mathrm{I}_i^* \mathrm{v}_i' \quad (8)$$

$$\mathrm{p}_0 = \mathrm{p}_0^* \quad (9)$$

The key thing to notice here is that p_i^* is expressed as a function of v_i', which is a local variable at link i. As a result, only propagation from leaf to base is required, and each parameter q_j does not affect terms computed for joints $j+1$ and beyond (Figure 3). Also note that p_i^* is in general not equal to p_i if $i \neq 0$. A term superscripted with an asterix should be treated only as an intermediary quantity, unless its subscript is zero in which case it is the desired aggregate result.

A linear time expression for the momentum Jacobian can be derived in a straightforward manner based on this form of the recursion (see [1]). Note that we are not simplifying or changing the outcome of the dynamics computation, only changing the order in which terms are computed. Aggregate momentum p_0 and the momentum Jacobian are exactly the same in both formulations.

2.2 Linear Time Force Jacobian

In a traditional inverse dynamics formulation, accelerations and forces are expressed as the time derivatives of Equations 5 and 6:

$$\mathrm{a}_i = \mathrm{X}_{i-1}^i \mathrm{a}_{i-1} + \mathrm{s}_i' \ddot{q}_i + \mathrm{v}_i \hat{\times} \mathrm{s}_i' \dot{q}_i \quad (10)$$

$$\mathrm{f}_i = \mathrm{X}_{i+1}^i \mathrm{f}_{i+1} + \mathrm{I}_i' \mathrm{a}_i + \mathrm{v}_i \hat{\times} \mathrm{I}_i' \mathrm{v}_i \quad (11)$$

where the symbol $\hat{\times}$ is the cross product operator for spatial vectors. As with momentum, this form results in an expression for the force Jacobian that requires $O(D^2)$ time to compute. For fast computation, we instead take the time derivative of Equation 8, which results in

$$\mathrm{f}_i^* = \mathrm{X}_{i+1}^i \mathrm{f}_{i+1}^* + \mathrm{v}_i' \hat{\times} \mathrm{p}_i^* + \mathrm{I}_i^* \mathrm{a}_i' + \dot{\mathrm{I}}_i^* \mathrm{v}_i' \quad (12)$$

This equation has the properties we are looking for. Velocity v_i' and acceleration a_i' are local to link i, and terms are propagated from leaf to base only. Note that as with aggregate momentum, f_i^* is in general different from the actual joint force f_i if $i \neq 0$.

Differentiating Equation 12 and accumulating the coefficients of derivative elements results in a compact expression for analytical derivatives, which are given in [1]. Each partial derivative of the aggregate force with respect to joint positions, velocities, and accelerations may be obtained in constant time, and the full Jacobian may be obtained in linear time.

3. Results

Optimal Motions. Figures 1 and 4 show a sampling of our results. Figure 1 shows a dismount. From top to bottom: initial motion, results with a flight time of $0.6s$, and results with a flight time of $0.8s$. Note the looser tuck and the higher flight trajectory in the $0.8s$ motion. The initial motion appears very unstable at landing. The character would fall over. This effect is eliminated in the optimization by enforcing the physics constraints of ground contact. This optimization required 26 seconds on a 750MHz Pentium 3 computer.

Figure 4 shows initial and final motion for a monkey bars example. This optimization required 2.4 minutes on a 750MHz computer. No touch-up was done on the results. In particular, the geometry of the monkey bars was not modeled. In this example, notice the swinging of the legs and arms, as well as body roll, pitch, and yaw. All of these effects are obtained as a result of the optimization process. In this example, the initial motion is rigid translation of the entire character.

Timing. To empirically test the advantage of our method for fast derivative computation, we ran a test example 5 times, each time with the identical setup except that a different technique was used to compute all required first derivatives. Figure 5 summarizes the results. The differentiation techniques tested were:

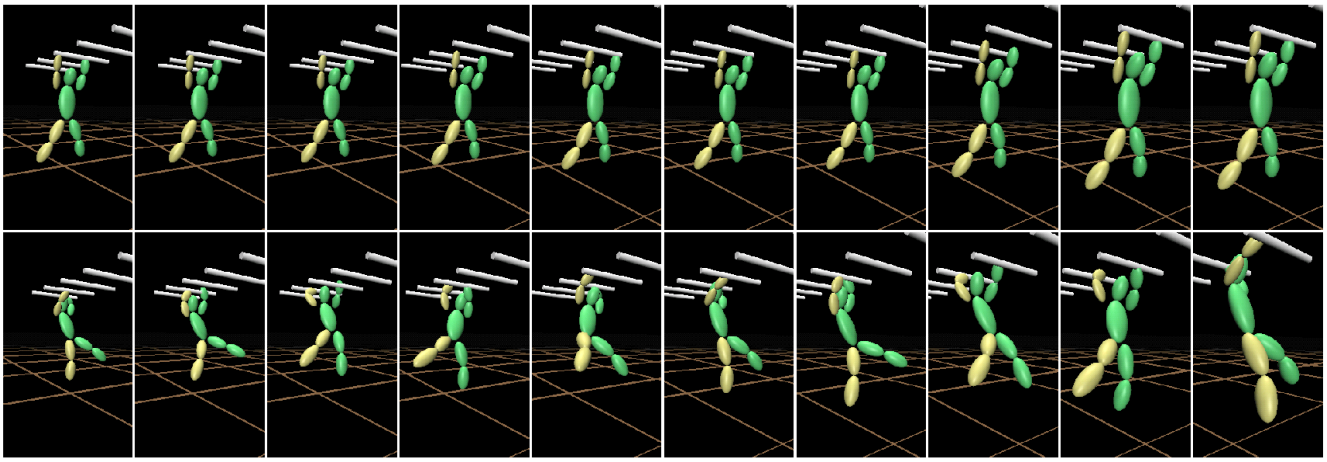


Fig. 4 Initial and optimized motion for a monkey bar example.

Technique	Time per iteration	Average % error
Our method	0.11s	0
Direct method	0.62s	0
NR1	0.97s	0.10
NR2	1.92s	1.0e-04
NR3	5.73s	1.5e-06

Fig. 5 Time required for one iteration using a variety of differentiation techniques.

- **Our method.** Analytical gradient computation using our approach.
- **Direct method.** Analytical gradient obtained by direct differentiation of the equations of motion.
- **NR1.** Numerical differentiation by ordinary forward differences.
- **NR2.** Numerical differentiation by central differences.
- **NR3.** Richardson-extrapolation of order 6.

4. Discussion

This paper contributes to physically based optimization by defining and exploring a restricted class of optimization problems where physics constraints are included and first derivatives of constraints and objective functions can be computed in linear time. The fact that first derivatives can be computed in linear time instead of quadratic time suggests that our problem is simpler than previous physically based approaches and similar in complexity to very successful kinematic approaches such as minimizing distance to a reference motion. We suspect that our solution landscape will be smoother than previous physically based optimization approaches, making it feasible to handle more complex characters.

For activities where joint torque limits are important, torque information must be taken into account to produce good results. An extreme example of this situation is the passive swing of a multi-link chain. Minimizing accelerations while maintaining physics constraints would produce a result that was valid for the body as a whole but would require non-zero torques at the joints—no whipping motion would be seen. Minimizing sum squared torques would produce the desired results. (Of course, truly passive motion can be created much more easily using forward dynamic simulation.)

More commonly, a limited set of torques or energy terms may be important. For example, the monkey bars motion appears to require very high torque at the waist. When physical parameters at certain joints are identified as important, our method can be extended to provide and differentiate these parameters for any K joints with running times of $O(KD)$, reaching the expected bound of $O(D^2)$ when all joint torques are required. An interesting research problem is to determine automatically when torques at a given joint should be considered.

Finally, we would like to emphasize that the main advantage of our approach may be as part of a more complete animation system. Our vision is that the ability to enforce physics constraints efficiently should be just one of the tools available to the animator. Details of the desired motion could be fleshed out using motion capture data, procedural techniques, keyframes, and/or objective functions appropriate to the specific task. We have shown that physics constraints can be enforced in an

efficient manner. Incorporating physics constraints into traditionally kinematic animation approaches is one direction of future work.

Acknowledgements This work was supported in part by NSF awards CCR-0093072 and IIS-0205224.

References

[1] A. C. Fang and N. S. Pollard. Efficient synthesis of physically valid human motion. In *SIGGRAPH 03 Proceedings*, 2003.

[2] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, Boston, MA, 1987.

[3] M. Gleicher. Motion editing with spacetime constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 139–148, Providence, RI, April 1997.

[4] M. Hardt, J. W. Helton, and K. Kreutz-Delgado. Optimal biped walking with a complete dynamical model. In *Proceedings of the 38th IEEE Conference on Decision and Control*, 1999.

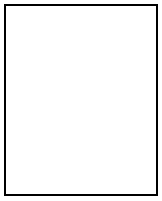
[5] C. K. Liu and Z. Popović. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH 02 Proceedings*, 2002.

[6] Z. Liu, S. J. Gortler, and M. F. Cohen. Hierarchical spacetime control. In *SIGGRAPH 94 Proceedings*, Annual Conference Series, pages 35–42. ACM SIGGRAPH, ACM Press, July 1994.

[7] M. G. Pandy and F. C. Anderson. Dynamic simulation of human movement using large-scale models of the body. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 2000.

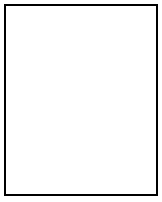
[8] Z. Popović and A. Witkin. Physically-based motion transformation. In *SIGGRAPH 99 Proceedings*, Annual Conference Series. ACM SIGGRAPH, ACM Press, August 1999.

[9] A. Witkin and M. Kass. Spacetime constraints. In J. Dill, editor, *Computer Graphics (SIGGRAPH 88 Proceedings)*, volume 22, pages 159–168, August 1988.



Anthony C. Fang

Anthony Fang is an Assistant Professor in the Department of Computer Science at the National University of Singapore. He received his PhD in Computer Science at Brown University in 2003. His primary research interest is in the synthesis of physically based animation of humanlike characters.



Nancy S. Pollard

Nancy Pollard is an Assistant Professor in the Robotics Institute and the Computer Science Department at Carnegie Mellon University. She received her PhD in Electrical Engineering and Computer Science from the MIT Artificial Intelligence Laboratory in 1994, where she performed research on grasp planning for articulated robot hands. Before joining CMU, Nancy was an Assistant Professor and part of the Computer Graphics Group at Brown University. Her primary research objective is to understand how to create natural motion for animated human characters and humanoid robots.