2006

Physics-Based Motion Retiming

James McCann
Carnegie Mellon University

Nancy S. Pollard
Carnegie Mellon University

Siddartha S. Srinavisa
Intel Research Pittsburgh

Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Physics-Based Motion Retiming

J. McCann, N. S. Pollard, and S. Srinivasa[2]

Carnegie Mellon University, Pittsburgh
[2] Intel Research, Pittsburgh

Abstract

By changing only the playback timing of a motion sequence, an animator can achieve a variety of effects that alter our perception of an event. In some scenarios, it may be important to consider physical properties of the motion when retiming (e.g., to preserve physical plausibility). However, existing retiming solutions can be quite time consuming when physical parameters are considered. This paper presents an interactive method for creating optimal motion retimings that takes into account physically based constraints and objective functions. We achieve fast performance through a precomputation phase where constraints are projected into the two-dimensional space of velocities and accelerations along the input motion path. Unlike previous approaches, our precomputation technique allows for rapid computation of plausible contact forces that result from retiming, and it also accommodates changing physical parameters. We demonstrate our approach by creating physically plausible results for changes in motion duration, manipulations of the gravity vector, and modifications of character limb masses.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling, Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism, Animation

1. Introduction

Captured human motion data is a rich source of natural human motion. However, it is important to have good techniques for editing this motion, as the user or an autonomous character will often have different goals from the director of the original motion capture sessions. A wide variety of editing techniques have been presented in the literature, ranging from interpolation and editing of key character poses, which can easily be performed interactively (e.g., [RCB98, LS99, Gle00]) to physically based optimization, which remains computationally expensive, despite dramatic improvements in efficiency (e.g., [SP05]). Safonova and Hodgins [SH05] have shown that kinematic techniques such as motion interpolation can in some cases preserve physical realism, but obtaining interactive performance with more general physically based optimization techniques remains difficult.

Motion retiming (Fig. 1) occupies an interesting place in the motion editing spectrum. When the character is constrained to go through a given set of poses and only the timing is unknown, the problem is sufficiently low dimensional that physically based objective functions and constraints can be incorporated into an interactive editing system. However, the space of solutions is also rich enough for a surprising variety of interesting and useful visual effects to be achieved. Good retiming techniques are needed to align motions, change the duration of a motion, or achieve a desired velocity at a given time. Retiming can also be used to exaggerate motion, to convey an appearance of strength or weakness, and to change a motion's style, for example to move from jerky, robotic behavior to a motion that is smooth and flows easily.

In this paper, we demonstrate that motion retiming with physically based objective functions and constraints can be performed in a highly interactive manner. The key to fast computation of optimal retimings is to precompute coefficients of an objective function and constraint boundaries as a function of the speed and acceleration of the character along the original path. This problem has been considered in robotics for retiming manipulator trajectories (e.g. Bobrow). We add to this body of work a technique for precomputing coefficients that allow us to obtain a plausible set

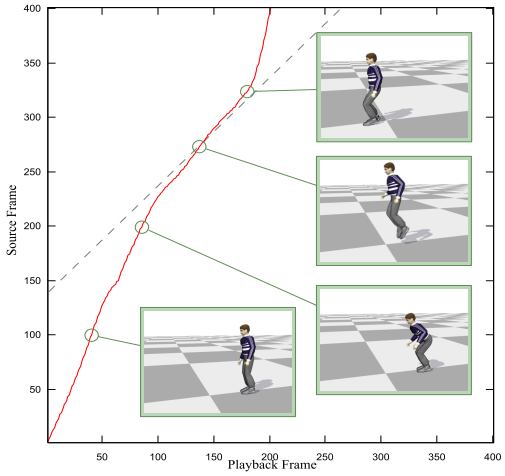

Figure 1: *A retiming of 16_06 from [CMU], constrained to finish in half the time. The flight phase has remained at normal speed (dotted line).*

of external forces when the character is in contact with its environment. These forces are needed for any constraints or objective function based on contact forces or joint torques, for example. We also extend this body of work to include precomputation of coefficients in situations where physical parameters such as masses of parts of the body may be subject to change.

We demonstrate our technique with examples of maintaining physical realism while changing the duration of the entire motion or the timing of key events. Additionally, we show that apparent physical properties of a character (limb mass) or environment (gravity) may be changed.

2. Background

There has been a great deal of interest in retiming existing motions, and we focus primarily on related work sharing this goal. In many applications, timing information is obtained directly from the end user. For example, commercial products such as Twixtor [Twi] give the user a great deal of flexibility in specifying playback timing for motion or video. In some cases, it may be more intuitive for a user to act out the desired timing, and a variety of interactive interfaces for timing control have been explored in the graphics literature (e.g., [BG95, DH00, DYP03, TBv04]). For example, Terra and Metoyer [TM04] separate specification of the animation path (the sequence of character poses) from specification of the timing along that path, and allow timing to be expressed interactively using a sketch-based interface. A different approach is proposed by Gomes and his colleagues [GVdG00], who make use of a frequency decomposition of the original motion and add or remove motion cycles to preserve frequency content when the desired duration of a cyclic motion is changed. When more than a single path is available, apparent timing can be controlled by interpolating between motions (e.g. [RCB98, KG04, GBT04]) or by having the character seamlessly transition between existing motion segments (e.g. [SSSE00, KPS03]). For example, Shödl and his colleagues [SSSE00] provide an interface that allows the user to control an actor's apparent running speed by manipulating a slider. Moving the slider results in a transition to a portion of the video texture where the actor is running at the speed desired.

User control of timing has also been used for physically-based systems. For example, van de Panne and his colleagues provide an interface that allows the user to specify timing for key events through keystrokes [Zv05, LvF00]. As the simulation runs, the user presses predefined keys to swap in new character setpoints or new controllers. Popović and his colleagues [PSE03] also allow user control over timing of a physical simulation. They demonstrate a system that takes as input a user sketch of a motion, including motion timing, and proceeds with an optimization step that adjusts the sketched motion to obtain physically plausible results.

The primary alternative to user control of timing is to generate a retiming that is optimal in some way. When physics of the motion are not considered, timewarping using dynamic programming is commonly used. This approach can provide a globally optimal alignment of motions to one another (e.g. [HPP05]) or to an external reference such as a musical beat (e.g. [KPS03]).

When physics is considered, dynamic timewarping can be very time consuming, because many repetitions of the inverse dynamics calculations must be run to execute the algorithm. Local techniques for physically based optimization (e.g., [WK88, SP05, LHP05]) could be used to timewarp the motion. However, even a local search will require many calculations of the inverse dynamics for the character. The main contribution of our paper is to show that optimal retimings for physically based objective functions and constraints can be performed extremely efficiently in the case where the character's path is given. The primary insight is that most of the constraints and objective functions that may be of interest can be projected onto the low dimensional space of state along the given path in an efficient precomputation phase. As a result, inverse dynamics can be run once per frame in a preprocessing phase and never considered again.

In robotics, similar problems of physically based timewarping have been considered for some time. Of primary interest has been time-optimal control [BDG85, SM85, SD89, CRDX88, PA05]. For this problem, it has been shown [BDG85] that the motion along a path that requires the smallest amount of time to execute is constructed from subsequences of maximum acceleration and deceleration. Therefore, the time-optimal control problem can be rephrased as one of finding switching points between these two extremes. Unfortunately, the broader class of objective

functions that we make use of in this paper has not been shown to have this property.

One of the key early findings from robotics used in our paper is that when the path is known, the dynamics equations of a robot can be expressed as a linear function of the acceleration and the square of velocity along the path [Hol84]. In other words, many physically based objective functions and constraints can be expressed in a form that is independent of character complexity.

This early work did not consider contact with the environment. Srinivasa and his colleagues [SEM05b,SEM05a] show that contact constraints can be expressed in the same simple form. However, the technique presented in their papers does not handle objective functions and constraints that require knowledge of joint torque and contact force values. We introduce a precomputation / interpolation approach to address this problem. In addition, we identify a broader class of objective functions that can be expressed as a linear function of acceleration and the square of velocity along the path. Finally, we note that the contribution of a wide variety of system properties such as limb masses can be expressed in this same form. Expressing these parameters in such a form allows the user to change these properties interactively during a retiming session.

3. Method

To perform a physically-based retiming we first compute the forces and constraints in effect on the system. We then use these forces to formulate an objective function which succinctly expresses the goal at hand. Finally, we use gradient descent and user interaction to find a locally optimal, satisfying to look at, cubic spline $p(t)$ which maps playback time into source time (see Fig.2).

3.1. Motion description

The dynamic motion of a character is governed by the Newton-Euler equations of motion. The equations describe the evolution of the configuration of the character q under the influence of external forces f_q and internal torques τ applied at each joint.

$$\mathsf{M}(q)\ddot{q}(t) + \dot{q}(t)^{\mathrm{T}}\mathsf{C}(q)\dot{q}(t) + f_q(t) = \tau(t) \tag{1}$$

where M and C are configuration dependent matrices.

Given an original motion $q(t)$, we wish to create a new motion by *retiming* the original motion, *i.e.*, assigning a time $p(t)$ for each $t \in [0,T]$, where T is the time to completion of the new (retimed) motion. However, to be physically valid, the retiming must satisfy the equations of motion (Eqn.1).

We shall now rewrite Eqn.1 in terms of the retiming p. The velocities at time $p(t)$ and at time t are related by the chain rule:

$$\frac{dq}{dt} = \frac{dq}{dp}\frac{dp}{dt} \tag{2}$$

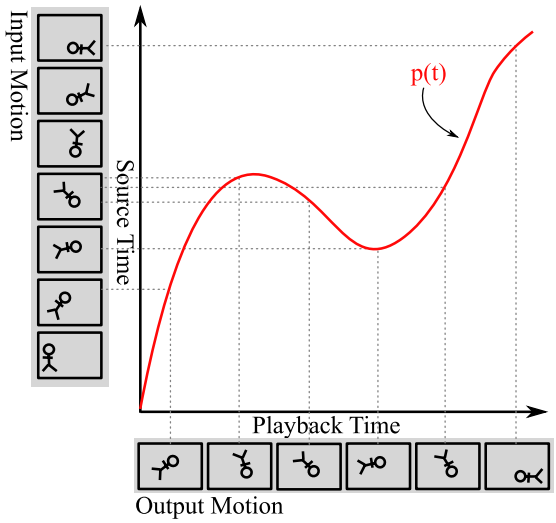

Figure 2: *The result of our retiming will be a cubic spline, $p(t)$, which will map from output time to input time. The output frame at time t is the input frame from time $p(t)$.*

For clarity, the above equation can be represented as:

$$\dot{q}(t) = q'\dot{p}(t) \tag{3}$$

where $'$ and $\dot{}$ refer to the derivatives w.r.t. p and t, respectively.

The accelerations at $p(t)$ and t are related by:

$$\ddot{q}(t) = q''\dot{p}(t)^2 + q'\ddot{p}(t) \tag{4}$$

At this juncture, it is important to state what is known, *i.e.*, what need not be freshly computed for each new retiming. We know the initial motion $q(t)$ and its time derivatives $\dot{q}(t)$ and $\ddot{q}(t)$. We also know q' and q'' since they are merely $\dot{q}(t)$ and $\ddot{q}(t)$ computed at time $p(t)$. This implies that any function of q, q' and q'' is known.

We shall henceforth drop the trailing "(t)" for p and its derivatives, for visual clarity.

Substituting Eqn.3 and Eqn.4 into Eqn.1, we obtain

$$\tau = \begin{bmatrix} 1 & \mathsf{M}q'' + q'^{\mathrm{T}}\mathsf{C}q' & \mathsf{M}q' \end{bmatrix} \begin{pmatrix} f_q & \dot{p}^2 & \ddot{p} \end{pmatrix}^{\mathrm{T}} \tag{5}$$

which can be rewritten as

$$\tau = \mathsf{K}(q,q',q'') \begin{pmatrix} f_q & \dot{p}^2 & \ddot{p} \end{pmatrix}^{\mathrm{T}} \tag{6}$$

For the purpose of fast computation, Eqn.6 is encouraging in two ways. Firstly, τ is linear in f_q, $\dot{p}^2$, and $\ddot{p}$ – something we use extensively below. Secondly, there is a nice decoupling of the constraint into terms involving the retiming and into the matrix K, which is a function solely of q, q' and q'' and is hence, by the argument above, a known function.

Instead of precomputing K analytically, we obtain coefficients in a manner that makes it trivial to add new variables such as limb mass to the system. Starting with a tested and working C++ implementation of Featherstone's recursive inverse dynamics method, we replaced the standard `double` data type with a custom generic polynomial type, and overloaded arithmetic operators for the necessary computations. Thus, by simply setting some of the input variables to include non-constant terms, we are able to compute the symbolic dependence of the joint torques and other quantities of interest on any parameter involved in the computation.

When new parameters are introduced, K may no longer be constant. However, for any given parameter setting (e.g., user selected limb mass), it can be quickly computed and will remain constant until the user explicitly changes another parameter.

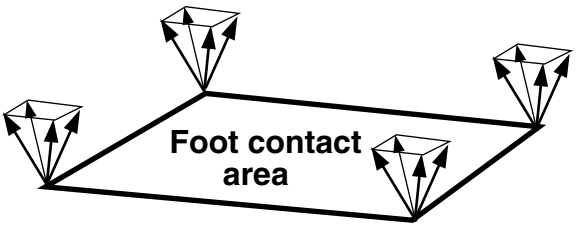

Figure 3: *One possible representation of legal forces at the foot is a positive linear combination of basis forces at the vertices of the polygon approximating the foot contact area. Any positive linear combination of the basis forces shown will meet ground contact constraints for that foot. For this example, the λ vector would contain magnitudes of force along each of the 16 basis directions.*

3.2. Motion constraints

The motion of the character is subject to a variety of constraints. We are specifically interested in constraints imposed by external forces acting on the character and constraints on the joint torques that the character can apply.

One source of external forces on the character are contacts with the world. For human motion, these contacts can be, for example, foot contact with the ground or contacts between the character and an object being manipulated. The reaction forces f_c that are produced at the contacts can be modeled by the laws of Coulomb friction which state that reaction force lies on or inside a friction cone $\mathcal{F}(\mu)$:

$$f_c \in \mathcal{F}(\mu) \tag{7}$$

where $\mathcal{F}$ is dependent on the coefficient of friction μ, a material property of the surfaces in contact.

In this paper, we approximate the friction cone with a conservative linear polyhedral convex cone (PCC) which is inscribed within the original cone, as illustrated for each of four contacts in Fig. 3. With this approximation in place, we can describe the friction cone as a set of forces that are positive linear combinations of a set of basis vectors:

$$\mathcal{F}(\mu) = \{\mathsf{P}(\mu)\lambda,\ \lambda \geq 0\} \tag{8}$$

A set of 16 basis vectors for contact with a rectangular foot, for example, is shown in Fig. 3.

The contact force f_c at any point in the simulation must be a positive linear combination of these basis vectors for the character to have plausible frictional contact with the world:

$$f_c = \mathsf{P}(\mu)\lambda,\ \lambda \geq 0 \tag{9}$$

Contact forces f_c are transformed to joint torques f_q using J^T, the transpose of the Jacobian, which depends on character pose q and maps external forces to joint torques:

$$f_q = \mathsf{J}^\mathsf{T}(q) f_c \tag{10}$$

The reason we denote the matrix as J^T is to be consistent with its usage in robotics literature.

Combining Eqn.9 and Eqn.10, we obtain:

$$f_q = \mathsf{J}^\mathsf{T}(q)\mathsf{P}(\mu)\lambda = \mathsf{N}(q,\mu)\lambda,\ \lambda \geq 0 \tag{11}$$

where N is a matrix that depends on character pose q and coefficient of friction μ. Eqn. 11 expresses the set of joint torques that can arise due to plausible external forces exerted at contacts with the environment.

The physical strength of the character limits the torques that can be applied at the joints. This constraint can be described as:

$$\tau \in [\tau_l, \tau_u] \tag{12}$$

where τ_l and τ_u are lower and upper bounds on the joint torques, respectively. While these vectors are time-invariant in our implementation, our framework easily admits configuration dependent torque limits.

3.3. Computing valid retimings

Given an original motion q, we can now restate the constraints for a physically valid motion in terms of constraints on the retiming p by combining Eqn.11, Eqn.12 and Eqn.6 as

$$\tau_l \leq \mathsf{K}(q,q',q'')\begin{pmatrix} \mathsf{N}(q,\mu)\lambda & \dot{p}^2 & \ddot{p} \end{pmatrix}^\mathsf{T} \leq \tau_u \quad \lambda \geq 0 \tag{13}$$

Eqn.13 describes a convex polytope in $[\ddot{p}, \dot{p}^2, \lambda]$ space. Note that, for an input motion and physical properties (encapsulated in K and N), this polytope needs to be calculated just once, and can be reused for all subsequent retimings. We do wish to compute the polytope quickly to facilitate interactive changing of physical properties.

Of course, since there will be several λ coefficients introduced for each contact, the dimensionality of this space is

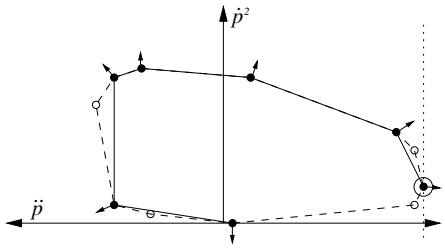

Figure 4: *The process of finding a hull in the $(\ddot{p}, \dot{p}^2)$ plane. The direction of maximization has been swept counterclockwise from straight down to right. The arrow at each point indicates the direction it was found at. The solid lines are the hull found so far, with the circled vertex the most recently added (the dotted line is perpendicular to the direction of maximization; notice no other vertex crosses it). The dashed lines are the true hull. Even though some vertices may be missed, the found hull will lie inside the true hull.*

far too high to construct the entire polytope. For the contacts shown in Fig. 3, for example, there would be 16 values of λ, one for each basis vector, along with one value each for $\dot{p}^2$ and $\ddot{p}$, making the polytope 18 dimensional. Indeed, we are primarily interested in the projection of the space onto the $\dot{p}^2, \ddot{p}$ plane, since this gives us boundaries on what timings are realistic.

To estimate this projection, we use the idea that each vertex of the projected polytope is of maximum extent in some direction. Therefore, our hull-finding procedure is as simple as picking some set of directions then calling a linear program solver to find the vertex of maximum extent in those directions, subject to the constraints in Eqn.13. The hull is refined by optimizing in different directions until a sufficient level of detail is obtained. An example is shown in Fig.4. This hull describes a boundary inside which all timings are feasible (for some λ values), and is thus useful for incorporation into our objective function.

3.4. Modeling contact forces for retimed motion

Having available the polytope in Fig. 4 allows us to search for retimings that meet joint torque and contact force constraints. Other constraints such as bounds on joint accelerations could be included in a similar fashion. However, we wish to find an optimal retiming, and for many objective functions computing the objective function itself requires knowing the λ values so that contact forces can be computed.

We already know feasible λ values at each vertex of the hull as an output of the optimization process that identified those vertices. In fact, because these are the extreme points on the hull, the λ values that are computed are the only ones possible at those points. Furthermore, because the constraints are linear, any interpolation of these extreme values to a point inside the hull will be a valid set of λ values. As a result, λ for any point in the hull could be estimated by triangulating the hull and using the barycentric coordinates of the desired $[\ddot{p}\dot{p}^2]$ point to interpolate between extreme λ values.

This simple interpolation scheme will always produce a valid solution, because all of the constraints are linear. However, in the interior of the hull, there is a family of solutions for λ, and so we take one extra step to ensure that values of λ near the original timing are reasonable. Specifically, we obtain a plausible set of λ values at a point inside the hull matching the original timing $[\ddot{p}, \dot{p}^2] = [0, 1]$. To do this, we set up a second optimization problem to minimize the summed absolute value of contact patch forces while meeting all constraints. This point then provides a center from which λ values are interpolated outward toward the extremes (see Fig.5 for a graphical representation).

The decision to use a single center point instead of a grid of points or no internal points at all was an engineering decision designed as a compromise between controlling the λ values in a principled way and adding too much extra storage. In examining sample slices we found that λ selected this way were within 20% of optimal close to the center point and within 60% over all plausible timings.

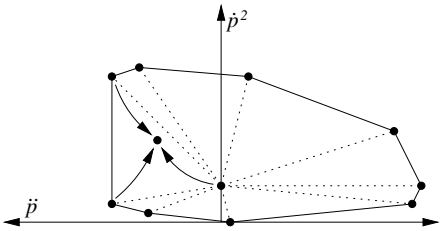

Figure 5: *Contact forces for a new point in the plausible region are interpolated from the containing triangle's corners.*

3.5. Objective Function

We would like an objective function that allows us to control both the plausibility of a motion and the 'look' of the motion. Consider the following examples:

- Changing the center of mass of a tumbling box. There is only one plausible timing for each COM and starting speed and we wish to find it.
- Halving the play time of a walking animation. While people may be physically able to walk twice as fast as normal, we would like to retain some of the 'look' of normal speed walking.
- Significantly altering the limb weight of a running figure. While the new figure may not have a plausible path, we are much more concerned with the 'look' of inertia that their heavier limb has.

- Finding the most efficient (minimal-force) timing of a keyframed run. In this case, we'd like the motion to 'look' as effortless as possible without becoming implausible.

An objective function for any of these examples can be expressed with a combination of torque matching and feasibility objectives. Consider the following objective function where c is the closest point in the feasible region to $[\ddot{p}, \dot{p}^2]$, τ' are target torque values, and α and β are weights.

$$\int_t \alpha \left([\ddot{p}, \dot{p}^2] - c \right)^2 + \beta \left(\tau(p(t), \dot{p}, \ddot{p}) - \tau'(p(t)) \right)^2 \quad (14)$$

The first term influences feasibility by pulling motions toward the feasible region (e.g., the polygon shown in Fig. 4). The second term allows us to control the 'look' of the motion by specifying a set of target torques. For example, torque minimization can be done by setting τ' to zero. The 'look' of another motion can be matched by setting τ' to the torques observed in that motion. Parameters α and β allow us to trade off between the feasibility and 'look' components.

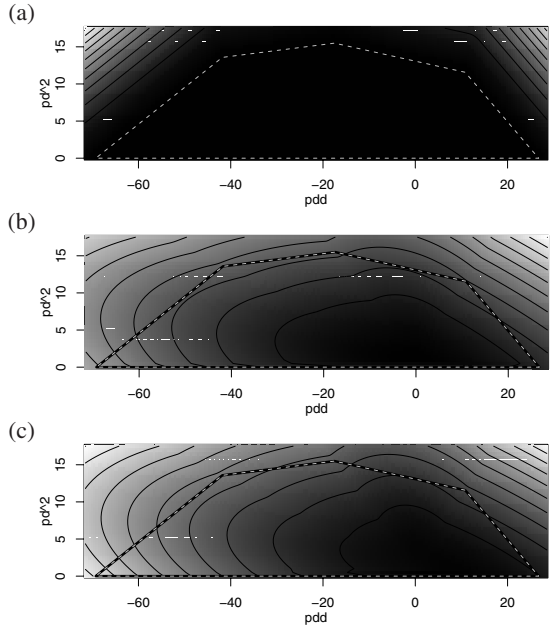

Figure 6: *Objective function component magnitude in the $\ddot{p} \times \dot{p}^2$ plane, evaluated at frame 143 of the 16_06 jump from [CMU]. The feasible region is superimposed. (a) is the feasible region distance. (b) is force matching to regular speed. (c) is force minimization.*

In practice, Eqn.14 is approximated by a recursive subdivision which samples at least once every quarter of a source frame. The integral of the objective function over time is obtained analytically by assuming that coefficients of the polynomial representation of the objective function are linearly interpolated between sample points.

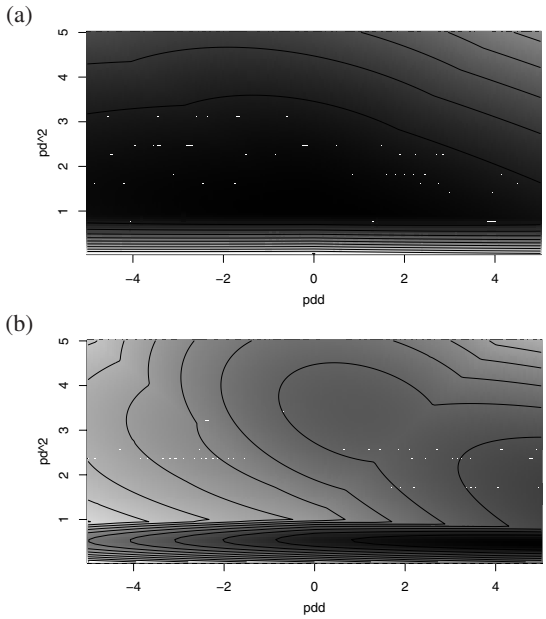

Figure 7: *Zoomed-in versions of the near normal speed regions of (a) the force matching and (b) the force minimization examples shown in 6. Notice that force minimization has many more minima than force matching.*

3.6. Interactivity

While it would be possible to find a global minimum of our objective function using a number of methods, we chose to favor a more user-friendly local optimization scheme (gradient descent).

Our prototype allows live interaction with the $p(t)$ function (represented as a cubic spline) during optimization, which lets users add, remove, adjust, and constrain control points as well as fight with or help the optimizer find a solution that optimizes the objective function and also suits the user's designs. In general, the gradient descent will reconverge in a few seconds after an adjustment is made.

Our interactive speed was achieved by realizing that, since our model of contact patch forces is (piecewise) linear in $\dot{p}^2$ and $\ddot{p}$, the force-matching portion of the objective may be reduced to a set of six coefficients per frame (those of $\dot{p}^4$, $\ddot{p}^2$, $\dot{p}^2\ddot{p}$, $\dot{p}^2$, $\ddot{p}$, and 1).

Additionally, because each point on the $p(t)$ spline is determined by only four control points, computing the partials at each control point is linear in the number of samples used when subdividing the objective function. (Note that as part of this process, we must locate the current triangle for the force model and also find the closest point c in the feasible region. However, both of these operations can be done in amortized constant time by taking advantage of query to query locality.)

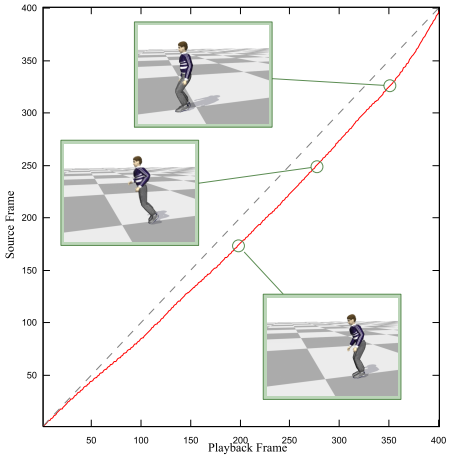

Figure 8: *A retiming of 16_06 from [CMU], with a point mid-flight constrained to play earlier. The solution moves the time distortion away from the highly dynamic portion of the motion.*

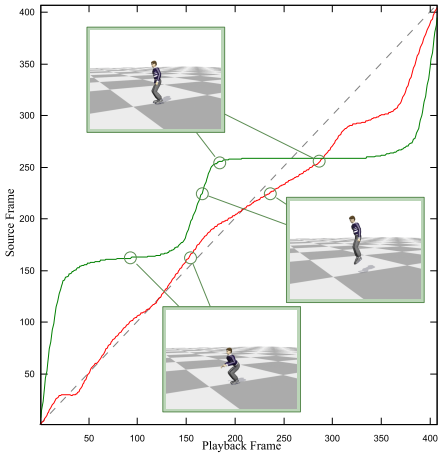

Figure 9: *Two retimings of 16_03 [CMU]. In red, lower gravity produces a longer flight time and slower takeoff. In green, higher gravity produces a shorter flight time and faster takeoff. (The plateaus in the high-gravity timing are due to the length being constrained.)*

4. Results

We applied our prototype system to changing the overall time of an animation, changing the mass of a body part, and changing gravitational strength and direction. We also performed a timing analysis to illustrate the dramatic effect of projecting all computations of interest into the space of parameters $\dot{p}^2$ and $\ddot{p}$.

For these examples we arrived at a mass distribution by modeling each segment approximately as a cylinder (and estimating their masses based on the idea that the human body has approximately the same density as water). Torque limits were determined by observing the joint torques over several actions as calculated without the aid of our contact model. The coefficient of friction was set at 0.5. A fully annotated skeleton file containing the exact values used is available with the electronic version of this paper.

4.1. Time Changes

We constrained the total playback length of a jump to half the original, setting the objective to match the original's forces and stay plausible (Fig.1). Our prototype was able to find idle time at the beginning and end of the jump which could be sped up without noticeable effect, leaving the flight phase intact at near 1x speed, since physically valid motion cannot be retimed in the ballistic phase.

We constrained a point at the center of the jumping motion to earlier in the animation, setting the objective to match forces and stay plausible (Fig.8). Our prototype again successfully located slack time in the clip and was able to satisfy the constraint without visibly warping the timing.

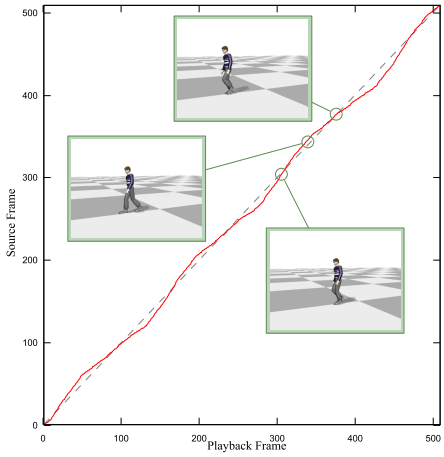

Figure 10: *A retiming of 16_16 from [CMU], with gravity pointing to the figure's left instead of down.*

4.2. Physical Parameter Changes

We changed the magnitude of gravity during an in-place jump, setting the objective to match forces and stay plausible (Fig.9). Our prototype successfully changed the jump timing to give the impression of both lower and higher gravity.

One need not constrain oneself to the magnitude of gravity. By setting gravity to point sideways in a walking motion, and using a force-matching objective, we caused the character to favor one leg (Fig.10).

We have also produced retimings to convey mass changes.

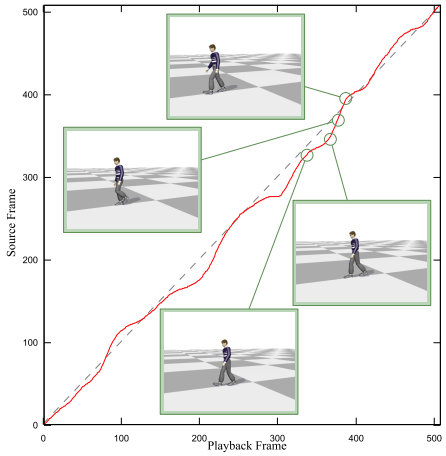

Figure 11: *A retiming of 16_16 from [CMU], with the left femur's weight multiplied by ten.*

	16_16	C10	C100	C1000
Force	66	33	170	1850
Feasible	12	5	20	250
Bake	0.78	x	0.037	0.21
Obj	0.0014	0.00027	0.00029	0.00029
NP-Obj	0.0044	0.00076	0.0046	0.050
Naive	78	38	190	2100

Figure 12: *Timing numbers (all in milliseconds per frame). Bake took too little time to measure for C10. Our pre-baked objective function (Obj) is four times faster than a per-dof function (NP-Obj), and several orders of magnitude faster than recomputing all quantities from scratch (Naive). Force, Feasible, and Bake give timings for important precomputation phases.*

Fig.11 shows a timing with the weight of the femur increased tenfold. The character favors positions where the heavy thigh is directly over the supporting (i.e. left) foot, since this reduces the torques at the knee.

4.3. Timing

Fig.12 gives timing results for several important processes over a number of motions. 16_16 is a walking motion, while C10, C100, and C1000 are chains with the indicated number of 3DOF links.

Forces gives the time to calculate the joint torques as a function of parameters. *Feasible* gives the additional time required to calculate the feasible region and set up the contact model. *Bake* is the additional time required to bake the objective function to a set of polynomial coefficients. *Obj* is time to evaluate the objective function. *NP-Obj* is the time to evaluate the objective function without the *Bake* step, which removes dependence on the number of dof from the computation. *Naive* would be time to re-compute everything from scratch each time the objective function is evaluated, and is provided as a point of comparison.

By evaluating the forces symbolically in terms of contact forces and timing, storing the feasible regions once computed, and baking the sum of per-dof polynomials in our objective into a per-frame polynomial, we save tremendous amounts of time and attain interactive convergence rates.

5. Discussion

We have demonstrated a method to change the timing of motions while retaining or modifying physical properties of the motion. By using a local optimization method and precomputation, our prototype achieved interactive frame rates. This method is suitable for inclusion in a real-time motion editing toolbox, and provides a physically principled method of performing clip length changes and temporal alignment as well as other, more complex, effects.

We also developed a linear-program-based approximate contact model which may be useful outside the context of changing motion timings. This model is suitable for use in any system where the forces applied to multiple contact areas such as the feet need to be resolved to a plausible solution quickly.

The primary benefits of our approach are its speed and the ability to incorporate physically motivated constraints and metrics. However, our system is limited in that ultimately only the timing of the motion can be changed. Some common edits require the character's overall path to change (e.g., modifying a motion to turn more sharply) or require the character's pose to change (e.g. a forward lean to accommodate a heavy backpack). Our system cannot accommodate these types of changes, as it is restricted to follow the path given as input by the user. However, if the user can first edit the motion by specifying new key poses, our system will then find the best timing of that new motion using physically based objectives and constraints, and we envision that a keyframe adjustment / retiming loop could be useful, especially for a novice animator keyframing a dynamic motion.

For future work, we believe that a great deal more flexibility could be obtained at little computational cost by adding just a few additional degrees of freedom to the system. For example, if modifications to a behavior can be described with a small number of basis directions, those basis directions could likely be added to our search space without sacrificing interactivity. One of the first modifications we plan to try is to add root position degrees of freedom to the optimization. The ability to change root position will allow us to create longer, higher jumps, bouncier runs, and change the translational motion of the root to achieve other user constraints and retiming goals. Our problem currently has a fairly smooth and well behaved search landscape, and we

are interested to see to what extent this remains true as additional degrees of freedom are added. If the search landscape becomes more complex, with many local minima, our interactive local technique may become less effective.

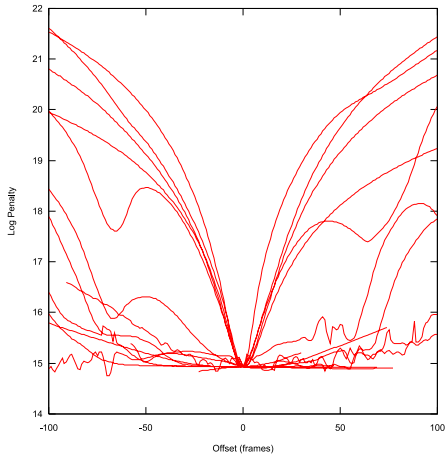

Figure 13: *The change in the penalty with respect to the offset of each control point from the local minimum penalty position.*

Even though we are using only a local optimization method, the minima that we find are quite good. Fig.13 shows how the value of the objective changes as individual control points are shifted. In general, the current timing curve lies in a large valley. In a few situations, however, we have observed nearby minima, which indicates that the optimization could terminate with a suboptimal retiming curve. However, there are generally few of these local minima, and the user is able to adjust the retiming curve by hand, e.g., to dislodge it into a better minimum, if the output motion is not desirable.

Our system requires estimating a number of parameters of physical system (torque limits, friction, mass distribution). While we found it sufficient to use a rough estimate of most quantities, it may be possible to generate most of these values by successive approximations based on additional input data, much as joint ranges are estimated based on captured range of motion exercises.

Alternatively, an effective force visualization framework that emphasizes violated constraints in a source motion would help users to create reasonable parameter estimates. Interesting future work may involve adding support for inverse dynamics and force visualization to animation software, thus giving animators another way to consider the validity of their motion.

Finally, we believe that our physically based motion retiming technique could be seamlessly incorporated into other applications where motion timing is controlled to achieve force or velocity related goals. For example, it could be used to manipulate motion where some specific end-effector velocity is desired for realism (for instance a penalty kick), or certain limb velocities are desired for smooth transitions (motion graphs). With good retiming tools, interactive games could even be created from just a few motion sequences and careful retiming to control, for example, the placement of a tennis or squash ball.

6. Acknowledgments

Alias/Wavefront provided the Maya software used to render the animations. Thanks to Moshe Mahler for the character models, and to Michael Erdmann and Matthew Mason for fruitful discussions on friction and dynamics. The motion capture data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. This work was also supported by NSF grants CCF-0343161, IIS-0326322, ECS-0325383, and CNS-0423546.

References

[BDG85] BOBROW J. E., DUBOWSKY S., GIBSON J. S.: Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research 4*, 3 (1985), 3–17.

[BG95] BALAGUER J., GOBBETTI E.: Sketching 3d animations. *Computer Graphics Forum 14*, 3 (1995), 241–258.

[CMU] CMU MOTION CAPTURE LIBRARY: http://mocap.cs.cmu.edu.

[CRDX88] CANNY J., REIF J., DONALD B., XAVIER P.: On the complexity of kinodynamic planning. In *IEEE Symposium on the Foundations of Computer Science* (1988), pp. 300–316.

[DH00] DONALD B. R., HENLE F.: Using haptic vector fields for animation motion control. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (2000), pp. 3435–3442.

[DYP03] DONTCHEVA M., YNGVE G., POPOVIĆ Z.: Layered acting for character animation. *ACM Transactions on Graphics (SIGGRAPH 03) 22*, 3 (2003), 409–416.

[GBT04] GLARDON P., BOULIC R., THALMANN D.: Pca-based walking engine using motion capture data. In *Proc. Computer Graphics International (CGI)* (2004).

[Gle00] GLEICHER M.: Comparative analysis of constraint-based motion editing methods. In *International Workshop on Human Modeling and Animation* (2000).

[GVdG00] GOMES J., VELHO L., DA SILVA F. W., GOLDENSTEIN S. K.: Motion processing using variable harmonic components. In *Proc. Computer Animation* (2000).

[Hol84] HOLLERBACH J. M.: Dynamic scaling of manipulator trajectories. *ASME J. Dynamic Systems, Measurement, Control 106*, 1 (1984), 102–106.

[HPP05] HSU E., PULLI K., POPOVIĆ J.: Style translation for human motion. *ACM Transactions on Graphics (SIGGRAPH 05) 24*, 3 (2005), 1082–1089.

[KG04] KOVAR L., GLEICHER M.: Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (SIGGRAPH 04) 23*, 3 (2004), 559–568.

[KPS03] KIM T., PARK S. I., SHIN S. Y.: Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics (SIGGRAPH 03) 22*, 3 (2003), 392–401.

[LHP05] LIU K., HERTZMANN A., POPOVIĆ Z.: Styles of human motion: Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (SIGGRAPH 05) 24*, 3 (2005), 1071–1081.

[LS99] LEE J., SHIN S. Y.: A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 1999* (1999), pp. 39–48.

[LvF00] LASZLO J. F., VAN DE PANNE M., FIUME E.: Interactive control for physically-based animation. In *Proceedings of SIGGRAPH 2000* (2000), pp. 201–208.

[PA05] PENG J., AKELLA S.: Coordinating multiple robots with kinodynamic constraints along specified paths. *International Journal of Robotics Research 24*, 4 (2005), 295–310.

[PSE03] POPOVIĆ J., SEITZ S. M., ERDMANN M.: Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics 22*, 4 (2003), 1034–1054.

[RCB98] ROSE C., COHEN M., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–40.

[SD89] SHILLER Z., DUBOWSKY S.: Robot path planning with obstacles, actuator, gripper, and payload constraints. *International Journal of Robotics Research 8*, 6 (1989), 3–18.

[SEM05a] SRINIVASA S. S., ERDMANN M. A., MASON M. T.: Control synthesis for dynamic contact manipulation. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)* (2005), pp. 2523–2528.

[SEM05b] SRINIVASA S. S., ERDMANN M. A., MASON M. T.: Using projected dynamics to plan dynamic contact manipulation. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)* (2005), pp. 3618–3623.

[SH05] SAFONOVA A., HODGINS J. K.: Analyzing the physical correctness of interpolated human motion. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2005).

[SM85] SHIN K., MCKAY N.: Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control 30* (1985), 531–541.

[SP05] SULEJMANPASIĆ A., POPOVIĆ J.: Adaptation of performed ballistic motion. *ACM Transactions on Graphics 24*, 1 (2005), 165–179.

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proceedings of SIGGRAPH 2000* (2000), pp. 489–498.

[TBv04] THORNE M., BURKE D., VAN DE PANNE M.: Motion Doodles: An interface for sketching character motion. *ACM Transactions on Graphics (SIGGRAPH 04) 23*, 3 (2004), 424–431.

[TM04] TERRA S. C., METOYER R. A.: Performance timing for keyframe animation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2004).

[Twi] TWIXTOR; RE:VISION EFFECTS, INC.: http://www.revisionfx.com/rstwixtor.htm.

[WK88] WITKIN A., KASS M.: Spacetime constraints. *ACM Transactions on Graphics (SIGGRAPH 88) 22*, 4 (1988), 159–168.

[Zv05] ZHAO P., VAN DE PANNE M.: User interfaces for interactive control of physics-based 3D characters. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)* (2005).