Synthesizing Physically Realistic Human Motion in Low-Dimensional, Behavior-Specific Spaces

Alla Safonova Jessica K. Hodgins Nancy S. Pollard

School of Computer Science
Carnegie Mellon University [*]

Abstract

Optimization is an appealing way to compute the motion of an animated character because it allows the user to specify the desired motion in a sparse, intuitive way. The difficulty of solving this problem for complex characters such as humans is due in part to the high dimensionality of the search space. The dimensionality is an artifact of the problem representation because most dynamic human behaviors are intrinsically low dimensional with, for example, legs and arms operating in a coordinated way. We describe a method that exploits this observation to create an optimization problem that is easier to solve. Our method utilizes an existing motion capture database to find a low-dimensional space that captures the properties of the desired behavior. We show that when the optimization problem is solved within this low-dimensional subspace, a sparse sketch can be used as an initial guess and full physics constraints can be enabled. We demonstrate the power of our approach with examples of forward, vertical, and turning jumps; with running and walking; and with several acrobatic flips.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation, G.1.6 [Numerical Analysis]: Optimization—Constrained Optimization

Keywords: physically based animation, optimization, motion capture

1 Introduction

Optimization methods are an easy and intuitive way to generate realistic motion. The user specifies a rough initial sketch of the motion and a set of constraints, such as start and end poses of the character or foot constraints. The optimizer then automatically finds a motion that "best" satisfies the user-specified constraints while preserving the physical validity of the motion.

Despite much research progress, this technique has not yet been shown practical for creating appealing animations when characters have a large number of degrees of freedom (DOF), when the motion must be physically correct, and when the initial guess is only a rough sketch. We are able to achieve this goal by exploiting the observation that most dynamic human motions are intrinsically low dimensional because they show a high degree of coordination. For example, six to eight dimensions adequately represent a human jump that appears quite similar to the original jump. This significant reduction in dimensionality is possible because the motion is coordinated; the legs and arms work together to generate the required velocity and control the landing and can therefore be represented as simple functions of just a few driving signals.

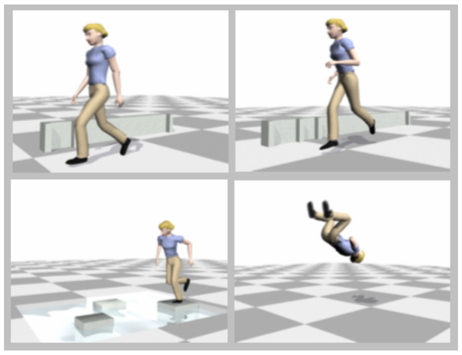
Figure 1: Synthesized motions: walking, running, jumping between stepping stones and a back flip.

We solve the optimization problem in a low-dimensional space by representing each frame of the desired motion as a linear combination of six to ten basis vectors. A simple dimensionality reduction technique, such as Principal Component Analysis (PCA), can be used to find this basis from examples of motions with similar behavior. Optimization is then used to find the linear coefficients that relate these vectors and produce the desired motion. The optimization problem is solved in the low-dimensional space, but the linear coefficients specify a physically valid motion for a full 60 DOF character. The solution often contains natural coordination patterns, an objective that is difficult to describe mathematically and that is usually not achieved when optimizing in the full-dimensional space.

Although optimization in the lower-dimensional space makes the problem tractable, constraints such as foot contact cannot always be satisfied exactly in this space. We solve this problem by using inverse kinematics to meet the constraints exactly while including a term in the optimization function that keeps the motion close to the low-dimensional basis. The optimizer then solves for a motion that is very close to the low-dimensional space, satisfies all the user-specified and physics constraints and minimizes such user-specified criteria as energy expenditure. We demonstrate the power of this approach through a number of different examples (Figure 1) and compare them to ground truth motion capture data.

One limitation of this approach is that a suitable set of motions must be selected to create the basis that will represent the optimized motion. We explore the effect of this choice on the ability to reconstruct a desired motion in Section 3 and explore the flexibility of a given basis set in Section 5.

2 Background

Constrained optimization techniques were first introduced to the graphics community by Witkin and Kass [1988]. They demonstrated the viability of this approach with a jumping Luxo lamp;

[*][alla|jkh|nsp]@cs.cmu.edu

its motion was quite compelling as it crouched in anticipation of a jump and compressed to absorb the impact. The user specified the start pose, end pose, and a physically based objective function; the optimizer computed the details of the motion. Despite this promising beginning, optimization has proven difficult for complex articulated characters, and subsequent research has focused on ways to make the approach viable for these more complex systems. Although no systematic studies have yet been published, the problem appears to be made more difficult by higher degree-of-freedom systems, physics constraints, torque-based optimization functions, and longer animations while domain knowledge in the form of existing control laws or motion data makes the problem more tractable.

One way to keep the problem tractable is to reduce the number of degrees of freedom. Popović and his colleagues [2000] developed a interactive system that gave the animator fine control over the motion of a single rigid body. Their system was able to produce a wonderful example of a hat spinning as it was tossed onto a hat rack. A number of researchers have shown that the freefall portion of a dive can be efficiently optimized for a simplified character [Liu and Cohen 1994; Crawford and Sastry 1995; Albro et al. 2000]. Huang and his colleagues [2000] computed the motion of characters of similar complexity performing such actions as weight lifting and pushups.

Simplifying a complex character also simplifies the problem: Popović and Witkin [1999] showed that significant changes to motion capture data can be made by manually reducing the character to the degrees of freedom most important for the task. The optimized motion was then mapped back up to the full character. Our approach is similar in that we also perform the optimization in a lower degree of freedom space, but we find that representation automatically and the motion we compute is physically correct for the full 60 DOF character.

Human motion with many degrees of freedom can be optimized when the animator provides closely spaced keyframes without exact timing information [Liu and Cohen 1995]. A related problem is dynamic filtering where an existing motion is optimized to make it physically realistic. In this formulation, the original motion can be thought of as a very closely spaced set of keyframes that function as soft constraints [Dasgupta and Nakamura 1999; Yamane and Nakamura 2000; Pollard and Reitsma 2001]. Short segments of motion can be computed for characters with many degrees of freedom as Rose and his colleagues [1996] demonstrated when they computed optimal transitions between human motion segments that began and ended with different but similar poses.

Physics constraints and an optimization function based on torque often make the problem more difficult to optimize. In contrast, purely kinematic techniques give the animator interactive control for making significant changes to the motion [Gleicher 1997; Lee and Shin 1999]. Simplified physical constraints also create tractable problems. Liu and Popović [2002] show that some dynamic effects can be preserved by enforcing patterns of linear and angular momentum that do not require the computation of such dynamic parameters as contact forces and joint torques.

If the dynamics of the system can be made more efficient, the search process also becomes more efficient. Fang and Pollard [2003] derived an algorithm for efficiently computing first derivatives of a broad range of physics constraints. Because their system never computed torques, they used a sum of weighted, squared accelerations as an optimization function. Grzeszczuk and his colleagues [1998] developed a neural network approximation of dynamics for a number of systems and used that approximation in the optimization step to reduce the computational cost of the gradient search. Outside of character animation, reduced order models of dynamics have been explored extensively in computer graphics and other fields (e.g., [Pentland and Williams 1989] [James and Fatahalian 2003] [Lall et al. 2003]), including for their use in speeding up optimization (e.g., [Ravindran 2000]). In our work, we do not simplify our representation of the dynamics of the system (inverse dynamics calculations are performed in the high-dimensional space); instead we reduce the dimensionality of the configuration space of the character that is explored during optimization. Simplification of dynamics could be added to our system with the goal of improving computation times still further.

One exception to the rule that higher-dimensional problems are intractable is the work of Pandy and Anderson [Anderson and Pandy 1999; Pandy and Anderson 2000]. Using months of computer time on a supercomputer, they found a pattern of muscle activations that minimized the metabolic energy consumed per unit distance travelled by the center of mass of the lower body for a single step of walking. Their human model had 10 segments and 23 degrees of freedom and was actuated by 54 muscles. The optimized motion provided an impressive match to muscle activation patterns recorded from human motion subjects.

In work done in parallel with that reported here, Sulejmanpašić [2004] showed that adaptation of ballistic motions with full physics is possible for high DOF characters if careful attention is paid to details such as proper variable scaling and initialization. Sulejmanpašić also explored using PCA on one or two captured motions to reduce the dimensionality of the search space, but did not find it to be effective, in part because reducing dimensionality below 16 DOF made it difficult to satisfy constraints during optimization. In the current paper, we show that by constructing a basis set from multiple examples of a behavior and by adding IK to the optimization process, a smaller number of DOF can be used successfully even without a good initial guess. We speculate that our use of PCA was successful while theirs was not because we used several examples of the desired behavior to create the basis while they used only one.

Finally, we note that many researchers in computer graphics, robotics, computer graphics, computer vision, machine learning and biomechanics have also explored the use of dimensionality reduction techniques to aid in clustering, modelling, and other processing of motion (see, e.g., [Jenkins and Mataric 2002] [Santello et al. 2002] [De la Torre and Black 2003] [Brand and Hertzmann 2000] [Li et al. 2002]).

3 Motivation

The algorithm proposed in this paper is based on two observations. First, many dynamic human motions can be adequately represented with only five to ten degrees of freedom. Second, motions with similar behavior can be used to construct a low-dimensional space that can represent well other examples of the same behavior.

Fifty to sixty dimensions are often used to represent a high quality human motion. For many behaviors, however, the movements of the joints are highly correlated. For example, during a walk cycle, the arms, legs and torso tend to move in a similar oscillatory pattern. As a result, the dimensionality of motions can be greatly reduced by an application of a simple dimensionality reduction technique such as PCA. Figure 2 shows the error between a number of motion sets and their projections onto optimal low-dimensional linear subspaces obtained using PCA. For the common human behaviors that we have tested, the error becomes small for representations with more than five to ten dimensions.

Figure 3 compares the error of representing forward jump motions in six different low-dimensional spaces. The error is averaged over twenty different jumping motions of varying height and length. The low-dimensional space computed from the motion that is being represented, (a), naturally provides the best representation. During the synthesis process, however, the desired motion is not known and thus this space cannot be computed. The low-dimensional space constructed from the motions of the same behavior (whether it is

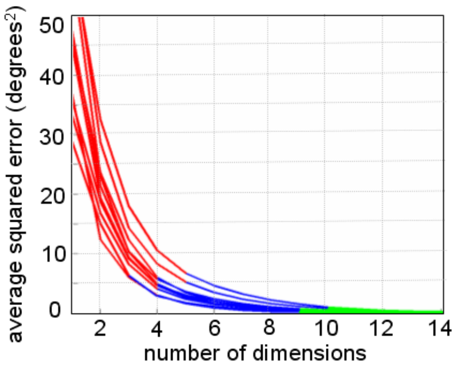

Figure 2: Error between a full-dimensional motion and the corresponding k-dimensional representation for a number of behaviors: running, walking, jumping, climbing, stretching, boxing, drinking, playing football, lifting objects, sitting down and getting up. The error was averaged over ten to twenty motions within each behavior. Each motion was represented as a collection of poses. The k-dimensional representation was computed by first using PCA to compute principal components for the set of full-dimensional poses and then projecting each full-dimensional pose onto the k components with the most variation in the data. The error is computed as an average squared error between the angles of the full-dimensional motion and its k-dimensional representation. The curve color gives an approximate measure of the visual quality: green indicates motions that look nearly indistinguishable from the full-dimensional motion; blue indicates motions that look very similar to the full-dimensional motion except for some sliding of the feet; and red indicates motions with large visual artifacts.

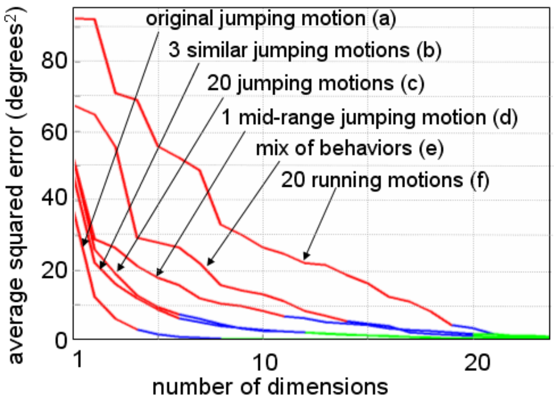

Figure 3: Motion representation error of a full-dimensional motion in a k-dimensional space averaged over twenty different jumping motions of varying height and length. For *each* of the motions the k-dimensional space is spanned by k principal components that are computed from: (a) the motion that is being represented, (b) three jumping motions that are visually similar to the motion being represented, (c) a set of twenty jumping motions, (d) a single mid-range jumping motion, (e) a mix of 150 behaviors (walking, running, jumping, climbing, punching, dribbling a basketball, lifting, drinking and other common human activities) and (f) twenty running motions. The sets of motions used for constructing spaces (c-f) are the same for each of the twenty motions. As in Figure 2, each curve is colored to indicate the visual quality.

just three similar motions as in (b) or a set of twenty motions as in (c)) can represent the motion quite well with seven dimensions. The low-dimensional space computed from one example of a given behavior, (d), requires higher dimensionality because it does not have enough generality to represent other motions well. When we incorporate motions with different behaviors, the required dimensionality of the space increases. A general mix of 150 motions, (e), however, provides a better representation than a behavior-specific representation for the wrong behavior, (f). Figure 3 illustrates the results of dimensionality reduction for forward jumping, but we have obtained similar results for other behaviors such as running and walking.

A seven-dimensional space constructed from several motions of the same behavior generally represents the desired motion well. When visual artifacts remain in this representation, they are usually the result of contacts. For example, during a forward jump, both feet are planted on the ground which constrains the relationship between the hip, knee, and ankle angles. This constraint may not be represented with sufficient accuracy in the low-dimensional space even though that space does model the basic fore/aft swinging of the arms and legs. We incorporate inverse kinematics as part of our optimization approach to reduce artifacts caused by this limitation (Section 4.2).

In Section 5, we demonstrate that a low-dimensional space constructed from a few motions of a particular behavior, such as spaces (b) and (c), can be used to synthesize physically realistic, natural-looking motions of same behavior but with quite different parameters (e.g., length, height or degree of turn for jumping or step length for running). A low-dimensional space constructed from one example of a particular behavior, such as space (d), may produce unnatural motions. A low-dimensional space constructed from motions with different behaviors (such as spaces (e) and (f) in Figure 3) requires solving an optimization problem of higher dimensionality which in our experience does not produce reliable results. In a higher-dimensional space, the problem is harder to solve and more dependent on having a "good" optimization criterion, something that is difficult to define mathematically.

4 Low-dimensional Optimization

To use our system, the animator specifies a rough sketch of the desired motion, M_s, and the constraints that should be enforced. The animator also selects the motions used to define the low-dimensional space for the desired motion. Constrained optimization is then used to automatically find a motion that minimizes some objective function subject to satisfying the user-specified and physics constraints (Figure 4).

We formulate the optimization problem by solving for the character's motion $M(t)$ as opposed to solving for a force function $T(t)$ where $T(t) = \{\tau_1(t)\ldots\tau_n(t)\}$ and $\tau_i(t)$ is the torque applied to joint i at time t. Inverse dynamics equations are used to compute the force function $T(t)$ at any given time t. A number of constraints must be set on the force function to preserve physical validity. The constraints are enforced at discrete times. The motion may be physically infeasible in between these points, but these constraints are enough to generate visually pleasing motion.

Because the unknown of the optimization problem, $M(t)$, is expressed in a low-dimensional space, it is easy to make this formulation of the optimization problem low-dimensional. Computing a reasonable initial guess for a motion $M(t)$ from an animator sketch is also easier than computing a reasonable torque function $T(t)$. Finally, inverse dynamics is usually easier to solve than forward dynamics [Featherstone 1987, page 79].

4.1 Low-dimensional Problem Representation

Each motion M consists of a sequence of frames $M(t) = \{p(t), Q(t)\}$, where $Q(t) = \{q_1(t)\ldots q_n(t)\}$ are the angles of all of

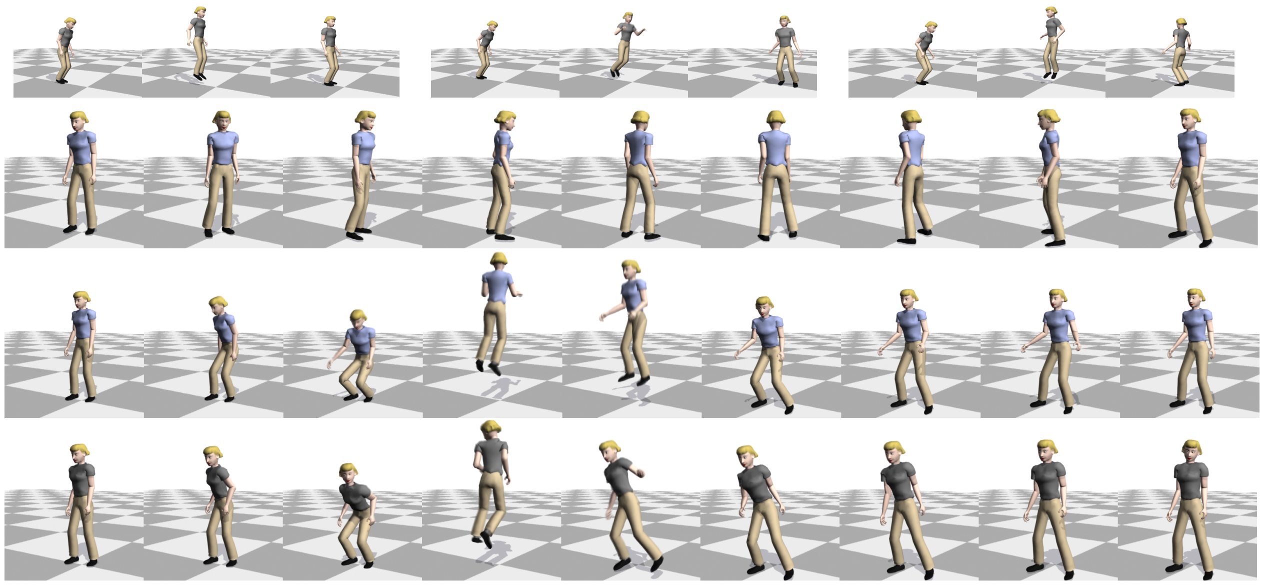
Figure 4: Synthesizing a vertical jump with a 360^o turn. (Top row) Motion capture data used to compute the low-dimensional space for the optimization: a forward jump, a forward jump with a 90^o turn, and a vertical jump with a 180^o turn. (Second row) Initial guess. Constraints were set on the first and the last pose of the motion and on the position of the feet during the stance phases. The duration of each stance phase and the desired height of the jump were also specified. (Third row) Synthesized motion. (Last row) Motion capture data of a similar motion for comparison.

the joints (including the root orientation) and $p(t)$ is the position of the root segment. $Q(t)$ is a point in an n-dimensional space. Let us consider a d-dimensional linear sub-space of the original space that is spanned by unit length orthogonal vectors, $B_1 \ldots B_d$, with origin $Q_m = (1/T)\sum_{i=1}^{T} Q(t_i)$, where T is the number of time samples. Then, we can approximate $Q(t)$ by a linear combination of the basis vectors $B_1 \ldots B_d$ using only d scalar coefficients $A_1(t) \ldots A_d(t)$, as:

$$Q'(t) = Q_m + B_1A_1(t) + B_2A_2(t) + \ldots + B_dA_d(t). \tag{1}$$

Principal Component Analysis (PCA) is a technique that finds $B_1 \ldots B_d$ so that the error $E = \sum_{i=1}^{T} (Q(t_i) - Q'(t_i))^2$ is minimized [Jolliffe 1986].

Given motions $M_{B1} \ldots M_{BK}$, selected by an animator, we use PCA to find the d-dimensional sub-space, L, that represents them with the smallest error E. If we now use Equation 1 for the representation of $M(t)$, then the unknowns of the optimization are root position $p(t)$, the mean of the joint angles Q_m, and the coefficients $A_1(t) \ldots A_d(t)$. We follow a standard approach of representing each $A_i(t)$ and $p(t)$ using cubic B-splines. The root position $p(t)$ is only unknown during the stance phase; during the flight phase the position of the center of mass (COM) of the character can be computed from the lift-off velocity, and $p(t)$ can then be computed from the COM position and the angles of the character.

The lower-dimensional space reduces the complexity of the optimization problem considerably. Let K be a number of control points in a B-spline curve used to approximate $p(t)$ and either $q_i(t)$ or $A_i(t)$. The full space has about $(n+3)K$ unknowns (where $n \approx 60$ for a human character), while the reduced dimensional space has $(d+3)K + n$ unknowns, with $7 < d < 9$ for the examples in this paper. Therefore, the number of unknowns is reduced by a factor of six to seven, which in our experience results in faster and more stable convergence of the optimization problem. In many cases, the mean of the joint angles, Q_m, can also be computed from example motions and excluded from the optimization function, further reducing the number of the unknowns. However, we kept Q_m as an unknown in the examples reported here.

Because we include root orientation in the dimensionality reduction analysis, we first preprocess all the motions by rotating them so that the character starts facing the positive Z direction. Keeping the root orientation in the basis worked well in the examples we tried, but for some motions with significant change in yaw orientation it may be better to treat root orientation as an additional variable and add 3K extra parameters to the optimization problem.

To estimate the dimensionality, d, of the linear sub-space L we use a standard heuristic [Fukunaga 1989]. We choose the smallest d such that:

$$E_r = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq 0.9 \tag{2}$$

where λ_i, $i = 1 \ldots n$, are the eigenvalues computed by PCA and sorted in decreasing order. Because the variance along the i-th principal component is given by the i-th eigenvalue, E_r is an indicator of how much information is retained when all data is represented in the optimal d-dimensional linear sub-space.

4.2 Inverse Kinematics for Limbs in Contact

In a low-dimensional space, the constraints that relate two or more points on the character's body may not be satisfied exactly. Consider, for example, the contact constraints on the character's feet during the double support phase of a jump. The low-dimensional space may not include a pose that would exactly satisfy constraints for both feet (or the pose may be unnatural). We address these problems by using inverse kinematics (IK) to transform the degrees of freedom in the optimization to a set that can be independently specified while maintaining constraints. IK is used to represent the angles for the arms or legs that are in contact with the environment. Intuitively, this allows an optimizer to satisfy contact constraints exactly by allowing the resulting motion to move slightly outside the space specified by the basis.

Consider a human arm consisting of three limb segments: upper arm, lower arm and hand. It can be represented by a seven DOF kinematic chain with one spherical (three DOF) joint at the shoulder, one at the wrist, and one revolute (one DOF) joint at the elbow.

When an arm is in contact with the environment (the position and the orientation of the hand segment is fixed) and the location of the shoulder joint is known there is a one DOF redundancy in the seven DOF kinematic chain representing the arm. As was pointed out by Korein and Badler [1982] and later by Lee and Shin [1999], this redundancy is in the "elbow circle" of the arm; the elbow can rotate even when the hand is in contact and the position of the shoulder joint is fixed. In this case all seven angles for the arm linkage can be analytically expressed through only one parameter, P, representing the elbow rotation (see [Tolani et al. 2000] for details).

Let $Q_u(t) = \{q_1(t)...q_k(t)\}$ be all the angles of the character joints excluding the joints that belong to the arms and legs that are in contact with the environment. As before, we represent these angles using a d-dimensional representation:

$$Q_u(t) = Q_m + B_1A_1(t) + B_2A_2(t) + \ldots + B_dA_d(t) \tag{3}$$

We represent the angles for all limbs (e.g., arms and legs) that are in contact with an environment, $Q_k(t) = \{q_{k+1}(t)...q_n(t)\}$ using IK:

$$Q_k(t) = F(P_1(t),...,P_r(t)) \tag{4}$$

where r is the number of limbs in contact, F is the IK function and $P_1(t)...P_r(t)$ are free parameters that define angles for the limbs in contact. Because there are four limbs (two arms and two legs), we are adding at most $4K$ unknowns to the optimization problem. When we use IK to compute some of the angles, the motion is no longer constrained to the low-dimensional space. To compensate for this, we add an additional term to the optimization function that favors poses in the low-dimensional space (see Section 4.4).

The model we use of the human leg (upper leg, lower leg, foot and toe segments) is similar in degrees of freedom to that of the arm except for the addition of the toe joint. If both toe and foot segments are constrained then the orientation of the foot segment needed for IK is known. If only the toe segment is constrained, then the angle between the toe and the foot segments is treated as part of $Q_u(t)$ and the foot segment orientation can be computed from this angle and the contact information given for the toe segment.

4.3 Constraints

Our system includes two types of constraints: user-defined constraints that allow the animator to control the resulting motion and constraints that ensure physical validity of the motion.

The most common user-specified constraints are pose constraints, contact constraints, and time constraints. We used pose constraints to fix initial, final, and key postures, such as a particular pose during a back flip. The constraint poses are specified in the full-dimensional space and then projected onto the low-dimensional space. We used contact constraints to specify foot configurations for ground contact, and time constraints were used to specify the durations of various phases of the motion. Some high-level user controls, such as height or length of a jump, speed of a walk or height of a back flip were also provided. If time was not provided and could not be computed from other constraints (e.g. height of a jump), then it was set as an additional variable in the optimization.

Constraints on the physical validity of the motion are added automatically by the system and are designed to preserve physical validity of the motion. They include joint angle limits, torque limits, and constraints on aggregate force. Joint angle limits are straightforward. Aggregate force constraints are set as in Fang and Pollard [2003], and include constraints for conservation of momentum during flight, as well as constraints on ground contact forces. For torque constraints, inverse dynamics allows us to compute torques for the character with a single point of contact with the environment. For the closed loop formed during multiple support phases, however, the problem is undetermined. We use the approximation method proposed by Ko and Badler [1996].

During implementation we found it beneficial to run the optimization problem in two steps: first, with no constraints on torque limits and aggregate forces; second, with these constraints added.

4.4 Objective Function

In our implementation, the objective function, $G(M)$, is a weighted sum of three components:

$$G(M) = w_T G_T(M) + w_A G_A(M) + w_P G_P(M) \tag{5}$$

The component $G_T(M)$ minimizes the sum of squared torques:

$$G_T(M) = \int \sum_{i=1}^{n} (\tau_i^2(t))dt \tag{6}$$

The component $G_A(M)$ ensures smoothness of the joint angle trajectories and root trajectory over time. It minimizes the sum of squared joint accelerations and sum of squared root accelerations:

$$G_A(M) = \int (\ddot{p}^2(t) + \sum_{i=1}^{n} \ddot{q}_i^2(t))dt \tag{7}$$

The component $G_P(M)$ ensures that the resulting motion has correlations between the angles and a distribution of poses around the mean pose similar to the ones found in the motions used to construct the basis. When we run PCA on motions $M_{B1} \ldots M_{BK}$, selected by an animator, we obtain both the principal components of the d-dimensional linear sub-space L, as well as the variance of the data points in these motions along the principal components, given by corresponding eigenvalues. The $G_P(M)$ component penalizes the deviation of coefficients from zero in inverse proportion to the standard deviation along the corresponding principal component:

$$G_P(M) = \int \sum_{i=1}^{d} (A_i^2(t)/\lambda_i)dt \tag{8}$$

From our experience, increasing the weight of the $G_T(M)$ component results in a more realistic motion, but the optimization problem takes longer to converge. Increasing the weight of the $G_A(M)$ component results in a faster convergence of the optimization problem, but the motion may not be energy efficient and as a result may not look as good. Increasing the weight of the $G_P(M)$ component results in a solution that more closely resembles the basis motions. Decreasing w_P, on the other hand, usually results in a solution that minimizes the sums of squared torques and accelerations better. The solution, however, may often look unrealistic (see Section 5). Optimizing in a low-dimensional space as well as using the $G_P(M)$ term results in natural coordination patterns.

4.5 Implementation Details

In our implementation we used a sequential quadratic programming package, SNOPT [1997], a commercially available library that solves general nonlinear constrained optimization problems. We also used a modeling language, AMPL [1989], that allows the user to easily formulate linear and nonlinear optimization problems in mathematical terms and automatically generates code appropriate for various solvers.

AMPL uses automatic differentiation to compute derivatives for the optimization function and the constraints. Automatic differentiation takes as input a section of code that computes the value of a function and outputs a new piece of code that computes analytical derivatives for that function. Unlike numerical differentiation methods, automatic differentiation is based on the chain rule computation of derivatives and is therefore considered an analytical differentiation method. It yields exact derivatives within machine accuracy.

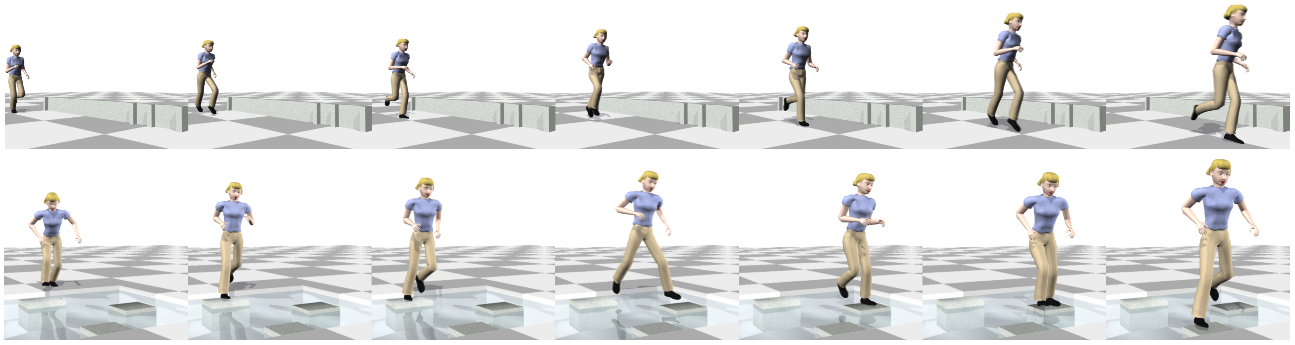

Figure 5: A forward run and a run across stepping stones.

5 Experimental Results

In this section we analyze the performance of our algorithm and show a number of motions generated for a human character with 60 degrees of freedom using our approach. For each example, an animator specified the start and end poses for the motion, the contact information and the timing for the stance and flight phases. The animator also selected a few motions from the database that contained similar behaviors to the desired motion. Our system then automatically found a motion that minimized the optimization function $G(M)$ (Equation 5) and satisfied the user-specified and the physics constraints. For all examples, the initial guess was a linear interpolation of the parameters (angles, position, IK parameters) for the starting and ending poses.

Each optimization took from three to sixty minutes to converge. The timing depends on many parameters: the length of the motion, the dimensionality of the low-dimensional space, the weighting on the energy component of the optimization function, $G_T(M)$, and the intrinsic parameters set for the optimizer (e.g., the number of major iterations, which we set to 1500). For example, when the problem was represented in a seven-dimensional space all the jumping motions took less than ten minutes to converge with equal weight set for both energy and smoothness components of the optimization function. Each jump was about two seconds in length. All the experiments were run on a 3GHz Pentium 4 computer with 1GB of RAM.

The bias toward "realistically" looking motions. We found that optimizing in a low-dimensional space as well as adding the $G_P(M)$ component to the optimization function biases the solution towards natural-looking motions. Optimizations run in higher-dimensional spaces with the weight on the $G_P(M)$ component set to zero usually result in a solution that minimizes the sum of squared torques and accelerations ($G_T(M)$ and $G_A(M)$ components) better, but has unnatural visual artifacts.

We generated the same forward jump three times: first, by representing the problem in a six-dimensional space with non-zero weight on the $G_P(M)$ component; second, by representing the problem in a twenty-dimensional space with the same weight on $G_P(M)$ component; third, by representing the problem in the same twenty-dimensional space with zero weight on $G_P(M)$ component. The solution that we obtained in the first and the second cases was natural looking, although running the optimization in higher dimensions was less reliable in general. The solution in the third case did not look natural, despite the fact that it was more optimal with respect to the $G_T(M)$ component. We observed similar results for other experiments.

The generality of the low-dimensional space. The same low-dimensional subspace can be used to synthesize a variety of motions. To demonstrate this, we used three different jumps to find a basis: a forward jump, a forward jump with a 90^o turn and a vertical jump with 180^o turn (Figure 4). We then synthesized a number of different jumps using that basis, varying the length of the jump and the size of the turn. The synthesized motion of the arms and legs is natural, and as the distance of the jump increases it appears as if the character tries harder. We were also able to synthesize a high vertical jump with a 360^o turn (Figure 4) although the basis only contained a low vertical jump with a 180^o turn. In a separate set of experiments, we also combined twenty forward jumps (the same jumps as were used in Figure 3) to compute a basis. According to equation 2 the dimensionality of the basis was eight. Jumps with varying height and length could also be synthesized in this space.

If the basis does not adequately represent the desired motion, the optimizer will produce a result that looks unnatural or violates the physics constraints. For example, we synthesized a long jump using a basis computed from a very short jump. The resulting motion looked physically realistic, but the arm motion was restricted in an unnatural way because the low-dimensional space did not allow sufficient arm motion. Synthesizing a jump using a basis computed from twenty running motions did not converge to a physically valid solution.

Other motions. We demonstrate the generality of our approach by synthesizing additional behaviors: running, running across stones, walking, and an acrobatic back flip.

Running (Figure 5). We used eight forward running motions and one motion of running across stepping stones to find the basis. We then synthesized three running motions with different step lengths, three motions that ran across stepping stones with varying placement and speeds and one running motion with a jump over water.

Walking (Figure 6). We used seven forward walking motions and one walking motion with the step over an obstacle to compute the basis. We synthesized three walks with different step lengths, including an exaggerated walk with a very large step length. We also synthesized two motions that stepped over obstacles of different heights.

Acrobatic Back Flip (Figure 7). We used one acrobatic back flip motion to compute the basis and synthesized two back flips of different distances and heights.

User Control. The optimizer finds a motion that minimizes a given objective criterion and is physically valid. Specifying additional constraints on the motion allows finer control over the details of the motion. We synthesized a back flip with straight legs in the middle of the flip by specifying a middle pose for the flip. We synthesized a run with a jump over water where the user controls the spread of the legs by adding a straddle pose in the middle of the jump. We also modified the style of the run over stepping stones by specifying two additional poses to generate a run where the feet are raised higher.

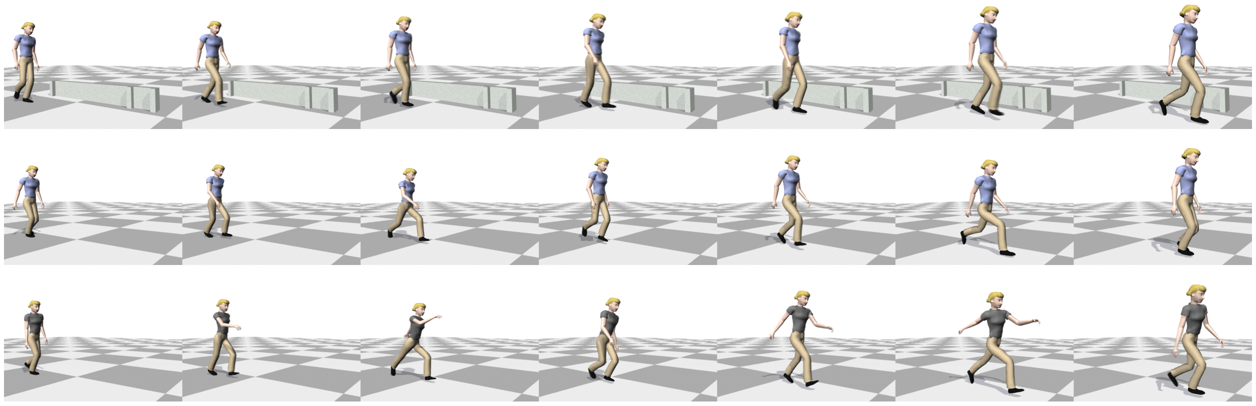
Figure 6: A normal walk, a walk with an exaggerated step length, and motion capture data of a walk with an exaggerated step length for comparison.

6 Discussion

The key insights in this paper are that optimization of human motion can occur much more effectively in a lower-dimensional space and that such a space can be easily created from motion capture clips of similar behaviors. The low-dimensional space allows us to generate natural-looking and physically valid motion for characters that have 60 degrees of freedom and only a rough sketch as the initial guess. This approach has proven to be effective for a wide variety of different human motions, both highly dynamic (back flip) and less dynamic (walking). The approach is quite robust to the choice of motion clips that are used to form the lower-dimensional space as long as they are not overly restrictive (e.g., a basis from just a single motion) and are similar behaviors to the desired behavior. It provides good control for the animator, allowing him or her to specify the location of the footfalls for a path of stepping stones or to specify an intermediate pose for the flight phase of a back flip.

Solving the optimization problem in an appropriate lower-dimensional space makes it not only more likely that the optimizer will converge to a feasible solution but also more likely that it will converge to a solution that matches the strategy a human would have used to perform that task. This feature of our approach reduces the burden on the animator because fewer constraints are required to guide the optimizer. To generate a walk motion, for example, Hardt and his colleagues had to specify symmetry and anti-symmetry constraints for human joint angles and contact forces [Hardt 1999]. The in- and out-of-phase motion of the walk was captured by our basis automatically, allowing us to generate a walk without explicitly specifying these constraints.

In the examples presented here, the animator selected the motions for the basis. This task was not burdensome and only required a few minutes of browsing a reasonably sized database (such as the publicly available database at mocap.cs.cmu.edu). However, it should be possible to search a database automatically for appropriate motions based on the constraints on the desired motion provided by the animator. Liu and Popović [2002] solved a related problem; their system searched a motion capture database for transition poses that separated constrained and unconstrained phases. They used the following training parameters: flight distance, flight height, previous flight distance, takeoff angle, landing angle, spin angle, foot speed at takeoff and landing, and the average horizontal speed. We believe that a similar technique could be used to search the database for the motions required to find the low-dimensional space. We have implemented a simpler algorithm that looks only at contact information and the overall direction of the motion to select motions from the database. We found that even this very simple approach generally selected appropriate motions.

7 Acknowledgments

We thank Moshe Mahler for his help in modeling and rendering. The authors were supported in part by NSF IIS-0196089, NSF EIA-0196217, NSF IIS-0205224, and NSF ECS-0325383.

References

ALBRO, J. V., SOHL, G. A., AND BOBROW, J. E. 2000. On the computation of optimal high-dives. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 3958–3963.

ANDERSON, F. C., AND PANDY, M. G. 1999. A dynamic optimization solution for vertical jumping in three dimensions. *Computer Methods in Biomechanics and Biomedical Engineering*, 2, 201–231.

BRAND, M., AND HERTZMANN, A. 2000. Style machines. In *Proc. of SIGGRAPH 2000*, 183–192.

CRAWFORD, L. S., AND SASTRY, S. S. 1995. Biological motor control approaches for a planar diver. In *Proc. IEEE Conference on Decision and Control*, 3881–3886.

DASGUPTA, A., AND NAKAMURA, Y. 1999. Making feasible walking motion of humanoid robots from human motion capture data. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 1044–1049.

DE LA TORRE, F., AND BLACK, M. J. 2003. A framework for robust subspace learning. In *Intl. Journal of Computer Vision*, vol. 54, 117–142.

FANG, A. C., AND POLLARD, N. S. 2003. Efficient synthesis of physically valid human motion. *ACM Trans. on Graphics 22*, 3, 417–426.

FOURER, R., GAY, D., AND KERNIGHAN, B., 1989. AMPL: A mathematical programming language.

FUKUNAGA, K. 1989. Statistical pattern recognition - 2nd edition. John Hopkins University Press, Baltimore.

GILL, P. E., MURRAY, W., AND SAUNDERS, M. A. 1997. SNOPT: An SQP algorithm for large-scale constrained optimization. Tech. Rep. NA–97–2, San Diego, CA.

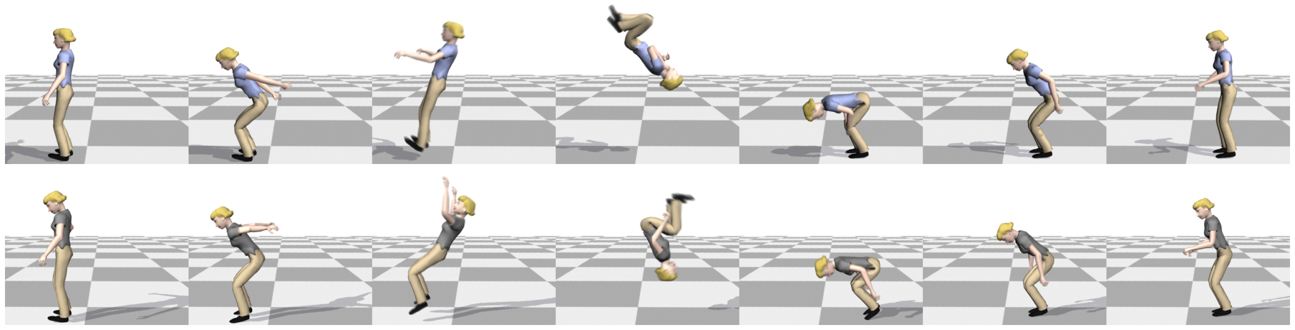
Figure 7: A back flip and motion capture data for comparison.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *Proc. of the 1997 Symposium on Interactive 3D Graphics*, 139–148.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proc. of SIGGRAPH 98*, 9–20.

HARDT, M. 1999. *Multibody Dynamical Algorithms, Numerical Optimal Control, with Detailed Studies in the Control of Jet Engine Compressors and Biped Walking*. PhD thesis, University of California San Diego.

HUANG, G., LO, J., AND METAXAS, D. 2000. Human motion planning based on recurive dynamics and optimal control techniques. In *Proc. of Computer Graphics International 2000*, 19–28.

JAMES, D. L., AND FATAHALIAN, K. 2003. Precomputing interactive dynamic deformable scenes. *ACM Trans. on Graphics*, 879–887.

JENKINS, O. C., AND MATARIC, M. J. 2002. Deriving action and behavior primitives from human motion data. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS)*, 2551–2556.

JOLLIFFE, I. 1986. Principal component analysis. Springer Verlag.

KO, H., AND BADLER, N. I. 1996. Animating human locomotion with inverse dynamics. *In IEEE Computer Graphics and Applications*, 50–59.

KOREIN, J., AND BADLER, N. 1982. Techniques for generating the goal-directed motion of articulated structures. 71–81.

LALL, S. P., KRYSL, P., AND MARSDEN, J. E. 2003. Structure-preserving model reduction for mechanical systems. In *Physica D 184*, 304–318.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proc. of SIGGRAPH 99*, 39–48.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: a two-level statistical model for character motion synthesis. *ACM Trans. on Graphics*, 465–472.

LIU, Z., AND COHEN, M. 1994. Decomposition of linked figure motion: Diving. In *5th Eurographics Workshop on Animation and Simulation*.

LIU, Z., AND COHEN, M. F. 1995. Keyframe motion optimization by relaxing speed and timing. In *6th Eurographics Workshop on Animation and Simulation*, 144–153.

LIU, C. K., AND POPOVIĆ, Z. 2002. Synthesis of complex dynamic character motion from simple animations. *ACM Trans. on Graphics 21*, 3, 408–416.

PANDY, M. G., AND ANDERSON, F. C. 2000. Dynamic simulation of human movement using large-scale models of the body. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 676–681.

PENTLAND, A., AND WILLIAMS, J. 1989. Good vibrations: Modal dynamics for graphics and animation. In *Computer Graphics (Proc. of SIGGRAPH 89)*, vol. 23, 215–222.

POLLARD, N. S., AND REITSMA, P. S. A. 2001. Animation of humanlike characters: Dynamic motion filtering with a physically plausible contact model. In *Yale Workshop on Adaptive and Learning Systems*.

POPOVIĆ, Z., AND WITKIN, A. P. 1999. Physically based motion transformation. In *Proc. of SIGGRAPH 99*, 11–20.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. P. 2000. Interactive manipulation of rigid body simulations. In *Proc. of SIGGRAPH 00*, 209–218.

RAVINDRAN, S. 2000. Reduced-order adaptive controllers for fluid flows using POD. In *Journal of Scientific Computing*, vol. 15, 457–478.

ROSE, C. F., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proc. of SIGGRAPH 96*, 147–154.

SANTELLO, M., FLANDERS, M., AND SOECHTING, J. F. 2002. Patterns of hand motion during grasping and the influence of sensory guidance. In *Journal of Neuroscience*, vol. 22, 1426–1435.

SULEJMANPAŠIĆ, A. 2004. *Adaptation of Performed Ballistic Motion*. Master's thesis, MIT Dept of EECS.

TOLANI, D., GOSWAMI, A., AND BADLER, N. 2000. Real-time inverse kinematics techniques for anthropomorphic limbs. In *Graphical Models 62 (5)*, 353–388.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *Computer Graphics (Proc. of SIGGRAPH 88)*, vol. 22, 159–168.

YAMANE, K., AND NAKAMURA, Y. 2000. Dynamics filter – concept and implementation of on-line motion generator for human figures. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 688–695.