A Hierarchical Approach to Interactive Motion Editing for Human-like Figures

Jehee Lee [†] Sung Yong Shin [‡]

Computer Science Department
Korea Advanced Institute of Science and Technology [§]

Abstract

This paper presents a technique for adapting existing motion of a human-like character to have the desired features that are specified by a set of constraints. This problem can be typically formulated as a spacetime constraint problem. Our approach combines a hierarchical curve fitting technique with a new inverse kinematics solver. Using the kinematics solver, we can adjust the configuration of an articulated figure to meet the constraints in each frame. Through the fitting technique, the motion displacement of every joint at each constrained frame is interpolated and thus smoothly propagated to frames. We are able to adaptively add motion details to satisfy the constraints within a specified tolerance by adopting a multilevel B-spline representation which also provides a speedup for the interpolation. The performance of our system is further enhanced by the new inverse kinematics solver. We present a closed-form solution to compute the joint angles of a limb linkage. This analytical method greatly reduces the burden of a numerical optimization to find the solutions for full degrees of freedom of a human-like articulated figure. We demonstrate that the technique can be used for retargetting a motion to compensate for geometric variations caused by both characters and environments. Furthermore, we can also use this technique for directly manipulating a motion clip through a graphical interface.

CR Categories: I.3.7 [Computer Graphics]: Three-dimensional Graphics—Animation; G.1.2 [Numerical Analysis]: Approximation—Spline and piecewise polynomial approximation

Keywords: Motion Editing, Motion Adaptation, Spacetime Constraints, Hierarchical Techniques, Inverse Kinematics

1 Introduction

Animating human-like characters is a recurring issue in computer graphics. Recently, motion capture has become one of the most promising technologies in character animation. Realistic motion data can be captured by recording the movement of a real actor with an optical or magnetic motion capture system. A motion library, that is an archive of reusable motion clips, is also commercially available. Much of the recent research in motion control has been devoted to developing various kinds of editing tools to produce a convincing motion from prerecorded motion clips. To reuse motion-captured data, animators often adapt them to a different character, i.e., retargetting a motion from one character to another [13], or to a different environment to compensate for geometric variations [3, 37]. Animators also combine two motion clips in such a way that the end of one motion is seamlessly connected to the start of the other [28].

The core of the motion manipulation can be modeled as a spacetime constraint problem [12, 13, 28]. Each kinematic constraint specifies the desired position or orientation of an end-effector, such as a foot and a hand, of an articulated figure at a specific time. The important features of the target motion are specified interactively as constraints, and the captured motion is deformed to satisfy those constraints.

Motion data consist of a bundle of motion signals. Each signal represents a sequence of sampled values for each degree of freedom. Those signals are sampled at a sequence of discrete time instances with a uniform interval to form a motion clip that consists of a sequence of frames. In each frame, the sampled values from the signals determine the configuration of an articulated figure at that frame, and thus they are related to each other by kinematic constraints. This structure yields two relationships among sampled values: *inter-frame* and *intra-frame relationships*. Through the use of an inverse kinematics solver, the intra-frame relationship, that is, the configuration of an articulated figure within each frame can be adjusted to meet the kinematic constraints. However, if each frame is considered independently, then there could be an undesirable jerkiness between consecutive frames. Therefore, we have to take account of the inter-frame relationship as well. For this purpose, we employ the multilevel B-spline fitting technique. We also present an efficient inverse kinematics algorithm which is used in conjunction with the fitting technique. Our approach is distinguished from the work of Gleicher [13] who addressed the same problem. He provided a unified approach to fuse both relationships into a very large non-linear optimization problem, which is cumbersome to handle. Instead, we decouple the problem into manageable subproblems each of which can be solved very efficiently.

Multilevel B-spline fitting techniques have been investigated to design smooth surfaces which interpolate scattered features within a specified tolerance [10, 20, 21, 34]. Among them, we extend the technique presented by Lee *et al.* [21] for adapting a motion to satisfy the constraints which are scattered over the frames. The multilevel B-splines make use of a coarse-to-fine hierarchy of knot sequences to generate a series of uniform cubic B-spline curves whose sum approaches the desired function. At each level in the hierarchy, the control points of the B-spline curve are computed locally with a least-squares method which provides an interactive performance. With this fitting technique, we cannot only manipulate a curve adaptively to satisfy a large set of constraints within a specified error tolerance, but also edit a curve at any level of detail to allow an arbitrary portion of the motion to be affected through direct manipulation. Exploiting these favorable properties of the

[†] jehee@jupiter.kaist.ac.kr, http://tclab.kaist.ac.kr/∼jehee
[‡] syshin@cs.kaist.ac.kr, http://tclab.kaist.ac.kr/∼syshin
[§] KAIST, 373-1 Kusung-dong Yusung-gu, Taejon, Republic of Korea

multilevel B-spline curves, we conveniently derive a hierarchy of displacement maps which are applied to the original motion data to obtain a new, smoothly modified motion. Because of this displacement mapping, the detail characteristics of the original motion can be preserved [3, 37].

The performance of our approach is further enhanced by our new inverse kinematics solver. It is commonplace to formulate the inverse kinematics with multiple targets as a constrained non-linear optimization for which the computational cost is expensive [28, 38]. As noticed by Korein and Badler [19], we can find a closed-form solution to the inverse kinematics problem for a limb linkage which consists of three joints, for example, shoulder-elbow-wrist for the arm and hip-knee-ankle for the leg. We combine this analytical method with a numerical optimization technique to compute the solutions for full degrees of freedom of a human-like articulated figure. Our hybrid algorithm enables us to edit the motions of a 37 DOF articulated figure, interactively.

The remainder of the paper is organized as follows. After a review of previous works, we give an introduction to the displacement mapping and the multilevel B-spline fitting technique in Section 3. In Section 4, we present our motion editing technique. In Section 5, we describe two inverse kinematics algorithms: One is designed to manipulate a general tree-structured articulated figure and the other is specialized to a human-like figure with limb linkages. In Section 6, we demonstrate how our technique can be used for interactive motion capture-based animation which includes adapting a motion from one character to another, fitting a recorded walk onto a rough terrain and performing seamless transitions among motion clips. Finally, we conclude this paper in Section 7.

2 Previous Work

2.1 Motion Editing

There have been an abundance of research results to develop motion editing tools. Bruderlin and Williams [3] showed that techniques from the signal processing domain can be applied to manipulating animated motions. They introduced the idea of displacement mapping to alter a motion clip. Witkin and Popović [37] presented a motion warping technique for the same purpose. Bruderlin and Williams also presented a multi-target interpolation with dynamic time warping to blend two motions. Unuma *et al.* [33] used Fourier analysis techniques to interpolate and extrapolate motion data in the frequency domain. Wiley and Hahn [35] and Guo and Robergé [14] investigated spatial domain techniques to linearly interpolate a set of example motions. Rose *et al.* [27] adopted a multidimensional interpolation technique to blend multiple motions all together.

Witkin and Kass [36] proposed a spacetime constraint technique to produce the optimal motion which satisfies a set of user-specified constraints. Brotman and Netravali [2] achieved a similar result by employing optimal control techniques. The spacetime formulation leads to a constrained non-linear optimization problem. Cohen [5] developed a spacetime control system which allows a user to interactively guide a numerical optimization process to find an acceptable solution in a feasible time. Liu *et al.* [22] used a hierarchical wavelet representation to automatically add motion details. Rose *et al.* [28] adopted this approach to generate a smooth transition between motion clips. Gleicher [12] simplified the spacetime problem by removing the physics-related aspects from the objective function and constraints to achieve an interactive performance for motion editing. He also applied this technique for motion retargetting [13].

2.2 Hierarchical Curve/Surface Manipulation

There is a vast amount of literature devoted to investigating hierarchical representations of curves and surfaces. Schmitt *et al.* [29] presented an adaptive subdivision method to produce a smooth surface from sampled data. Forsey and Bartels [9] introduced a hierarchical B-spline representation to enhance surface modeling capability. This representation allows details to be adaptively added to the surface through local refinement. They also employed the hierarchical representation for fitting a spline surface to the regular data sampled at grid points [10]. Welch and Witkin [34] proposed a variational approach to directly manipulate a B-spline surface with scattered features, such as points and curves. Lee *et al.* [20, 21] suggested an efficient method for interpolating scattered data points. They also demonstrated that image warping applications can be cast as a surface fitting problem by adopting the idea of displacement mapping. Although authors used different terms, such as hierarchical and multilevel B-spline surfaces, to refer to their hierarchical structures, their underlying ideas are the same, that is, a coarse-to-fine hierarchy of control lattices. Another class of approaches is due to multiresolution analysis and wavelets. Finkelstein and Salesin [8] used B-spline wavelets for multiresolution editing of curves. Many authors have investigated multiresolution analysis for manipulating spline surfaces and polygonal meshes [4, 6, 23, 30].

2.3 Inverse Kinematics

Traditionally, inverse kinematics solvers can be divided into two categories: *analytic* and *numerical solvers.* Most industrial manipulators are designed to have analytic solutions for efficient and robust control. Kahan [16] and Paden [24] independently discussed methods to solve an inverse kinematics problem by reducing it into a series of simpler subproblems whose closed-form solutions are known. Korein and Badler [19] showed that the inverse kinematics problem of a human arm and leg allows an analytic solution. Actual solutions are derived by Tolani and Badler [32].

A numerical method relies on an iterative process to obtain a solution. Girard and Maciejewski [11] addressed the locomotion of a legged figure using Jacobian matrix and its pseudo inverse. Koga *et al.* [18] made use of results from neurophysiology to achieve an "experimentally" good initial guess and then employed a numerical procedure for fine tuning. Zhao and Badler [38] formulated the inverse kinematics problem of a human figure as a constrained non-linear optimization problem. Rose *et al.* [28] extended this formulation to handle variational constraints that hold over an interval of motion frames.

3 Preliminary

3.1 Displacement Mapping

The configuration of an articulated figure is specified by its joint angles in addition to the position and orientation of the root segment. We will denote the position of the root by a 3-dimensional vector and the others by unit quaternions. It is well-known that unit quaternions can represent 3-dimensional orientations smoothly and compactly without singularity [31]. This representation can also describe a human joint conveniently.

A motion is a time-varying function which provides the configuration of an articulated figure at a time. We denote a motion by $\mathbf{m}(t) = (\mathbf{p}(t), \mathbf{q}^1(t), \cdots, \mathbf{q}^n(t))^T$, where $\mathbf{p}(t) \in \mathbb{R}^3$ and $\mathbf{q}^1(t) \in \mathbb{S}^3$ describe the translational and rotational motion of the root segment, and $\mathbf{q}^i(t) \in \mathbb{S}^3$ gives the rotational motion of the $(i-1)$-th joint for $2 \leq i \leq n$.

A displacement map (also called a warp function) describes the difference between two motions [3, 37]. Gleicher [13] provided a good explanation to introduce this technique into a spacetime formulation. In our mathematical setting, the displacement map is defined as $\mathbf{d}(t) = \mathbf{m}(t) \ominus \mathbf{m}_0(t)$, where $\mathbf{d}(t) = (\mathbf{v}^0(t), \cdots, \mathbf{v}^n(t))^T$ and $\mathbf{v}^i(t) \in \mathbb{R}^3$ for $0 \leq i \leq n$. Thus, a new

motion can be obtained by applying the displacement map to the original motion as $\mathbf{m}(t) = \mathbf{m}_0(t) \oplus \mathbf{d}(t)$, that is,

$$\begin{pmatrix} \mathbf{p} \\ \mathbf{q}^1 \\ \vdots \\ \mathbf{q}^n \end{pmatrix} = \begin{pmatrix} \mathbf{p}_0 \\ \mathbf{q}_0^1 \\ \vdots \\ \mathbf{q}_0^n \end{pmatrix} \oplus \begin{pmatrix} \mathbf{v}^0 \\ \mathbf{v}^1 \\ \vdots \\ \mathbf{v}^n \end{pmatrix} = \begin{pmatrix} \mathbf{p}_0 + \mathbf{v}^0 \\ \mathbf{q}_0^1 \exp(\mathbf{v}^1) \\ \vdots \\ \mathbf{q}_0^n \exp(\mathbf{v}^n) \end{pmatrix}. \quad (1)$$

Here, $\exp(\mathbf{v})$ denotes a 3-dimensional rotation about the axis $\frac{\mathbf{v}}{\|\mathbf{v}\|} \in \mathbb{R}^3$ by angle $\|\mathbf{v}\| \in \mathbb{R}$. We refer the readers to Kim *et al.* [17] for details of quaternion algebra and the exponential map. With the displacement mapping, we are able to deal with both position and orientation data in a uniform way; the displacement map is a homogeneous array of 3-dimensional vectors, while the configuration of an articulated figure is represented as a heterogeneous array of a vector and unit quaternions.

3.2 Multilevel B-spline Approximation

Lee *et al.* [21] proposed a multilevel B-spline approximation technique for fitting a spline surface to scattered data points. In this section, we give a brief summary to introduce their fitting technique. Since we need to manipulate a curve rather than a surface, our derivation focuses on curve fitting.

Let $\Omega = \{t \in \mathbb{R} | 0 \le t < n\}$ be a domain interval. Consider a set of scattered data points $\mathcal{P} = \{(t_i, x_i)\}$ for $t_i \in \Omega$. To interpolate the data points, we formulate an approximation function f as a B-spline function which is defined over a uniform knot sequence overlaid on the domain Ω. The function $f(t) = \sum_{k=0}^{3} B_k(t - \lfloor t \rfloor) b_{\lfloor t \rfloor + k - 1}$ can be described in terms of its control points and uniform cubic B-spline basis functions B_k, $0 \le k \le 3$. Here, b_i is the i-th control point on the knot sequence for $-1 \le i \le n + 1$. With this formulation, the problem of deriving function f is reduced to that of finding the control points that best approximate the data points in $\mathcal{P}$.

Since each control point b_j is influenced by the data points in its neighborhood, we can define the *proximity set* $\mathcal{P}_j = \{(t_i, x_i) \in \mathcal{P} | j - 2 \le t_i < j + 2\}$ which affect the value of b_j. Simple linear algebra using pseudo inverse provides a least-squares solution

$$b_j = \frac{\sum_{(t_i, x_i) \in \mathcal{P}_j} \omega_{ij}^2 \beta_{ij}}{\sum_{(t_i, x_i) \in \mathcal{P}_j} \omega_{ij}^2} \quad (2)$$

which minimizes a local approximation error $\sum_{(t_i, x_i) \in \mathcal{P}_j} \|f(t_i) - x_i\|^2$. Here, $\omega_{ij} = B_{j+1-\lfloor t_i \rfloor}(t_i - \lfloor t_i \rfloor)$ comes from a B-spline basis function, and $\beta_{ij} = \frac{\omega_{ij} x_i}{\sum_{k=0}^{3} B_k(t_i - \lfloor t_i \rfloor)}$ takes an effect to pull the curve toward a data point (t_i, x_i).

There is a trade-off between the shape smoothness and accuracy of the approximation function. If the knot spacing of the approximation function is too coarse, it may leave large deviations at the data points. Conversely, if the function is defined over an excessively fine knot sequence, its shape would undulate through the data points. Multilevel B-spline approximation uses a series of B-spline functions with different knot spacings to achieve a smooth shape while accurately approximating given data points in $\mathcal{P}$. The function from the coarsest knot sequence provides a rough approximation, which is further refined in accuracy by the functions derived from subsequent finer knot sequences.

A multilevel B-spline function is a sum of cubic B-spline functions $f_0, \cdots, f_h$ which are defined over uniform knot sequences overlaid on domain Ω. The knot sequences yield a coarse-to-fine hierarchy. Without loss of generality, we assume that the knot spacing of f_i is coarser than that of f_j for any $i < j$. The multilevel B-spline approximation begins by determining the control points of f_0 with Equation (2), which serves as a smooth initial approximation. f_0 may leave a deviation $\Delta^1 x_i = x_i - f_0(t_i)$ for each point (t_i, x_i) in $\mathcal{P}$. The next finer function f_1 is used to approximate the difference $\mathcal{D}_1 = \{(t_i, \Delta^1 x_i)\}$. Then, the sum $f_0 + f_1$ yields a smaller deviation $\Delta^2 x_i = x_i - f_0(t_i) - f_1(t_i)$ for each point (t_i, x_i). At a level k of the hierarchy, function f_k is derived to approximate $\mathcal{D}_k = \{(t_i, \Delta^k x_i)\}$ where $\Delta^k x_i = x_i - \sum_{l=0}^{k-1} f_l(t_i)$. By repeating this process to the finest level, we can incrementally derive the final approximation function $f = \sum_{k=0}^{h} f_k$.

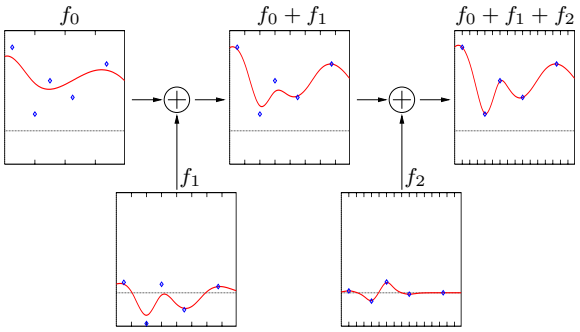


Figure 1: Hierarchical curve fitting to scattered data through multilevel B-spline approximation

4 Hierarchical Motion Editing

4.1 Basic Idea

Given the original motion $\mathbf{m}_0$ and a set $\mathcal{C}$ of constraints, our problem is to derive a smooth displacement map $\mathbf{d}$ such that a target motion $\mathbf{m} = \mathbf{m}_0 \oplus \mathbf{d}$ satisfies the constraints in $\mathcal{C}$. Current motion editing techniques represent the displacement map as an array of spline curves defined over a common knot sequence [13, 37]. Each spline curve gives the time-varying motion displacement of its corresponding joint. With a finer knot sequence, we can possibly find a solution that accurately satisfies all the constraints in $\mathcal{C}$. However, we have to pay a higher computational cost for the accuracy due to the finer knot sequence. Ideally, we wish to determine the density of knots to yield just enough shape freedom for an exact solution. However, the target motion is not known in advance and thus we require the displacement map which allows details to be added by adaptively refining the knot sequence.

We adopt the hierarchical structure [21] reviewed in Section 3.2 to perform this adaptive refinement. The multilevel B-spline approximation technique was employed to derive a warp function for image morphing and geometry reconstruction. In our context, we extend this technique to handle motion data. From the displacement map $\mathbf{d}$, we derive a series of successively finer submaps $\mathbf{d}_1, \cdots, \mathbf{d}_h$ that give the corresponding series of incrementally refined motions, $\mathbf{m}_1, \cdots, \mathbf{m}_h$.

$$\mathbf{m}_h = (\cdots((\mathbf{m}_0 \oplus \mathbf{d}_1) \oplus \mathbf{d}_2) \oplus \cdots \oplus \mathbf{d}_h). \quad (3)$$

Here, $\mathbf{d}_k$, $1 \le k \le h$, is represented by an array of cubic B-spline curves in the 3-dimensional vector space. The component curves of $\mathbf{d}_k$ are defined over a common sequence τ_k of knots that are uniformly spaced. The knot sequences τ_k, $1 \le k \le h$, form a coarse-to-fine hierarchy. τ_1 is placed on the coarsest level in the hierarchy and τ_h is on the finest level. The motion $\mathbf{m}_1 = \mathbf{m}_0 \oplus \mathbf{d}_1$ provides a rough approximation to a target motion, which is further refined by applying $\mathbf{d}_2$ to give a more accurate approximation $\mathbf{m}_2$. By applying the submaps one by one, we can incrementally obtain the final motion. Be aware that it is a very frequent mistake to have

$\mathbf{d} = \sum_{k=1}^{h} \mathbf{d}_k$ from an erroneous derivation, that is, to substitute $\exp(\mathbf{v}_1)\exp(\mathbf{v}_2)\cdots\exp(\mathbf{v}_h)$ for $\exp(\mathbf{v}_1+\mathbf{v}_2+\cdots+\mathbf{v}_h)$ in Equation (1) and (3). This derivation is not correct, since the quaternion multiplication is not commutative.

4.2 Constraints

To specify the desired features of the target motion, two categories of constraints are employed: The ones in the first category are used to describe an articulated figure itself, such as a joint limit and an anatomical relationship among joints. Those in the other category are for placing end-effectors of the figure at particular positions and orientations which are interactively specified by the user or automatically derived from the interaction between the figure and its environment. For example, we first specify the contact point between the foot and the ground through a graphical interface and automatically modify the point later in accordance with the geometric variation of the ground. We assume that a constraint in either category is defined at a particular instance of time. A variational constraint that holds over an interval of motion frames can be realized by a sequence of constraints for the time interval. An ordered pair (t_j, C_j) specifies the set C_j of constraints at a frame t_j.

4.3 Motion Fitting

In order to compute a displacement map, it is necessary to estimate the motion displacement of each joint at every constrained frame. The displacement of the joint at a particular frame is interpolated by the corresponding component curve of the displacement map and thus smoothly propagated to the neighboring frames. Suppose that we are now at the k-th level for $1 \le k \le h$. At each constrained frame t_j, our inverse kinematics solver gives the configuration $\mathbf{m}^{t_j}$ of the character, that meets a given set of constraints (t_j, C_j). Since there may exist many possible configurations that satisfy all constraints in C_j, we consistently choose the one that is minimally deviated from the motion $\mathbf{m}_{k-1}$ at the previous level. That is, we minimize

$$\begin{aligned} \|\mathbf{d}^{t_j}\|_\alpha^2 &= \|\mathbf{m}^{t_j} \ominus \mathbf{m}_{k-1}(t_j)\|_\alpha^2 \\ &= \sum_{i=0}^{n} \alpha_i \|\mathbf{v}^i\|^2, \end{aligned} \tag{4}$$

where $\mathbf{d}^{t_j} = (\mathbf{v}^0, \cdots, \mathbf{v}^n)$. We can control the rigidity of an individual joint by adjusting its weight value.

Combining the inverse kinematics solver with the hierarchical structure given in Equation (3), we give the following motion fitting algorithm:

Algorithm 1 Hierarchical motion fitting

INPUT : the original motion $\mathbf{m}_0$, the set $\mathcal{C}$ of constraints
OUTPUT : a new motion $\mathbf{m}_h$

1: **for** $k := 1$ **to** h **do**
2: $\quad\mathcal{D} := \emptyset$
3: $\quad$**for each** $(t_j, C_j) \in \mathcal{C}$ **do**
4: $\quad\quad\mathbf{m}^{t_j} :=$ IK_solver$(C_j, \mathbf{m}_{k-1}(t_j))$
5: $\quad\quad\mathbf{d}^{t_j} := \mathbf{m}^{t_j} \ominus \mathbf{m}_{k-1}(t_j)$
6: $\quad\quad\mathcal{D} := \mathcal{D} \cup (t_j, \mathbf{d}^{t_j})$
7: $\quad$**end for**
8: $\quad$Compute $\mathbf{d}_k$ by curve fitting to $\mathcal{D}$
9: $\quad\mathbf{m}_k := \mathbf{m}_{k-1} \oplus \mathbf{d}_k$
10: **end for**

This algorithm evaluates $\mathbf{d}_k$, $1 \le k \le h$, in the coarse-to-fine order. At each level k in the hierarchy, we compute the motion displacement $\mathbf{d}^{t_j} = \mathbf{m}^{t_j} \ominus \mathbf{m}_{k-1}(t_j)$ for all (t_j, C_j) in $\mathcal{C}$ using the inverse kinematics solver (See lines 2–7). Here, $\mathbf{m}_{k-1}$ is either the original motion ($k = 1$) or has been already obtained in the previous level ($k > 1$). The newly computed displacements in $\mathcal{D} = \{(t_j, \mathbf{d}^{t_j})\}$ are used as the keyframes for deriving the displacement map $\mathbf{d}_k$(See line 8). Several techniques are available for computing $\mathbf{d}_k$ that interpolates the displacements. One possible solution is to employ an iterative numerical method that may give an optimal solution. However, we choose a curve fitting method given in Equation (2) to achieve an interactive performance. Our method is extremely fast, since the solution can be obtained analytically unlike the numerical method that normally requires a heavy computational cost for iterations. In general, local curve fitting with B-splines may have several drawbacks. The resulting curve may be less accurate and could have undulations because of the lack of the global propagation of a displacement. Fortunately, our hierarchical structure can compensate for such drawbacks by globally propagating the displacement at a coarse level and performing the later tuning at fine levels.

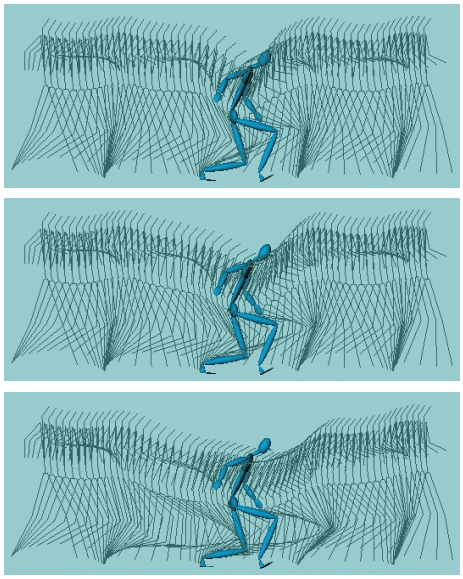


Figure 2: A live-captured walking motion was interactively modified at the middle frame such that the character bent forward and lowered the pelvis. The character is depicted at the modified frame. The range of deformation is determined by the density of the knot sequences. The knots in τ_1 are spaced every (top) 4, (middle) 6, and (bottom) 12 frames.

4.4 Knot Spacing

For simplicity, our implementation doubles the density of a knot sequence from one level to the next. Therefore, if τ_k has $(n+3)$ control points on it, then the next finer knot sequence τ_{k+1} will have $(2n+3)$ control points. The density of a knot sequence τ_k, $1 \le k \le h$, determines the range of influence of a constraint on the displacement map at a level k. This is of great importance for direct manipulation. For example, consider the situation that we interactively adjust the configuration of an articulated figure by dragging one of its segments at a certain frame through a graphical interface. The user input is interpreted as constraints which are immediately added to the set of prescribed constraints. Then, our system smoothly deforms a portion of the motion clip around this modified frame. Here, the range of influence on the motion clip is mainly dependent on the spacing of τ_k. The larger spacing between

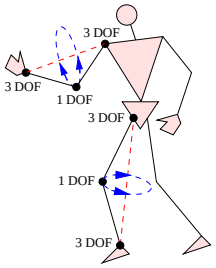


Figure 3: A human-like figure that has explicit redundancies at its limb linkages

knots yields the wider range of deformation (See Figure 2). Therefore, the displacement map $\mathbf{d}_1$, that is derived from the coarsest sequence τ_1, has non-zero values over the widest range to smoothly propagate the change of the motion. The subsequent finer displacement maps $\mathbf{d}_k$, $2 \leq k \leq h$, perform successive tunings to satisfy the constraints.

The density of the finest knot sequence τ_h controls the precision of the final motion $\mathbf{m}_h$. If τ_h is sufficiently fine to accommodate the distribution of constraints in the time domain, $\mathbf{m}_h$ can exactly satisfy all constraints. However, our algorithm may leave a small deviation for each constraint in $\mathcal{C}$ even with several levels in the hierarchy. In our experiments, we need just four or five levels for visually pleasing results, that can be further enhanced to achieve an exact interpolation by enforcing each constraint independently with the inverse kinematics solver.

4.5 Initial Guesses

For a spacetime problem, a good initial guess on the desired solution is very important to improve both the convergence of numerical optimization and the quality of the result [13]. We obtain an initial guess for motion fitting by shifting the position of the root segment in the original motion. To motivate this, consider the walking motion that is adapted to the rough terrain as shown in Figures 5 (a) and (b). The position of a stance foot, that touches the surface of the terrain, is pulled upward at a small hill on the terrain, and thus the character is unwantedly forced to squat. Even though the inverse kinematics solver tries to minimize the deviation of joint angles, it cannot prevent the knee bending completely. To reduce this artifact, we change the position of the root segment due to the change of geometry. Specifically, we displace the root segment by the average shift of the contact positions at each frame. The shift of the root segment position at a frame can also be smoothly propagated to the neighboring frames using the multilevel B-spline fitting method.

5 Inverse Kinematics

The most time consuming component in the motion fitting algorithm is the inverse kinematics solver which is invoked very frequently at each level of the fitting hierarchy. Therefore, the overall performance of a hierarchical fitting critically depends on the performance of the inverse kinematics solver. We describe, in this section, two inverse kinematics algorithms. In Section 5.1, we introduce an inverse kinematics algorithm for a general tree-structured figure with spherical joints based on numerical optimization techniques. In Section 5.2, we present a faster specialized algorithm for a human-like figure with limb linkages. The latter algorithm combines the numerical techniques with an analytical method illustrated in Section 5.3.

5.1 A Numerical Approach

Our inverse kinematics solver is based on a constrained non-linear optimization technique that minimizes the objective function subject to a set C of constraints. We have an additional burden to enforce the unitariness of each quaternion parameter $\mathbf{q} = (w, x, y, z)$. One possible solution would be to augment the set C with a new constraint $w^2 + x^2 + y^2 + z^2 - 1 = 0$ for the unit quaternion parameter. We circumvent this extra constraint from the observation that every orientation $\mathbf{q}$ can be expressed as a rotation $\exp(\mathbf{v}) \in \mathbb{S}^3$ from a fixed reference orientation $\mathbf{q}_0 \in \mathbb{S}^3$, that is, $\mathbf{q} = \mathbf{q}_0 \exp(\mathbf{v})$. Therefore, we can parameterize $(\mathbf{p}, \mathbf{q}^1, \cdots, \mathbf{q}^n)$ by a simple vector $\mathbf{x} = (x_0, \cdots, x_{3n+2}) \in \mathbb{R}^{3n+3}$ using the displacement $\mathbf{d} = (\mathbf{v}^0, \cdots, \mathbf{v}^n)$ from a given reference configuration $(\mathbf{p}_0, \mathbf{q}_0^1, \cdots, \mathbf{q}_0^n)$ as follows:

$$\begin{aligned} \mathbf{p} &= \mathbf{p}_0 + \mathbf{v}^0, \quad \text{and} \\ \mathbf{q}^i &= \mathbf{q}_0^i \exp(\mathbf{v}^i), \quad 1 \leq i \leq n, \end{aligned} \tag{5}$$

where $\mathbf{v}^i = (x_{3i}, x_{3i+1}, x_{3i+2})$ for $0 \leq i \leq n$. Letting the reference configuration be $\mathbf{m}_{k-1}(t_j)$ in Equation (4), we reduce the objective function to a quadratic form of $\mathbf{x}$. Accordingly, our constrained optimization problem is formulated as follows:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{M}\mathbf{x}, \\ \text{subject to} \quad & c_i(\mathbf{x}) = 0, \quad i \in N_e, \\ & c_i(\mathbf{x}) > 0, \quad i \in N_i, \end{aligned}$$

where $\mathbf{M}$ is a diagonal matrix that determines the rigidity of individual parameters in $\mathbf{x}$.

A typical approach to the constrained optimization is to transform the constrained problem into an unconstrained version with extra parameters (the Lagrange multipliers) or extra energy terms (penalty functions). We avoid illegal configurations by employing the penalty method that allows us to handle the equality and inequality constraints in a uniform way [25]. The objective function for the unconstrained version is

$$g(\mathbf{x}) = f(\mathbf{x}) + \sum_{i \in N_e} \omega_i c_i(\mathbf{x})^2 + \sum_{i \in N_i} \omega_i (\min(c_i(\mathbf{x}), 0))^2, \tag{6}$$

where ω_i weights each individual constraint. We adopt the conjugate gradient method to minimize this function [26].

5.2 A Hybrid Approach

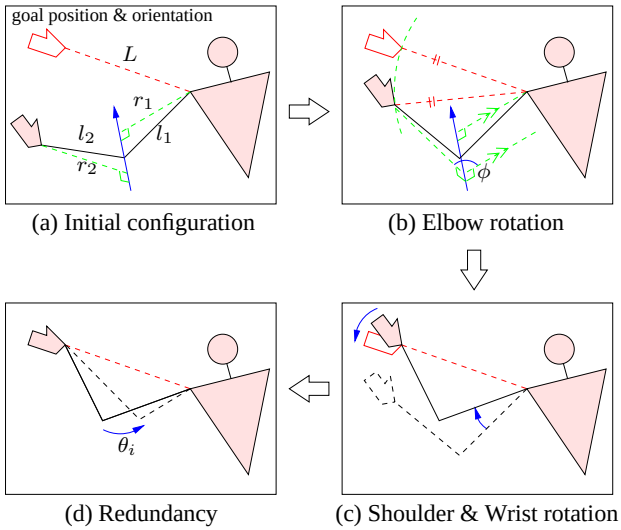
The major difficulty of solving an inverse kinematics problem stems from the excessive DOFs of an articulated figure. A reasonable human model may have about 40 DOFs for computer animation, while we specify much fewer constraints for manipulating the figure. For a figure of n DOFs, we can remove c of those DOFs with a set of c independent constraints imposed on it. The remaining $(n - c)$ DOFs span the solution space of the problem.

A reduced-coordinate formulation parameterizes the redundant DOFs with a reduced set of $(n - c)$ variables. One explicit redundancy in the human body is the "elbow circle" that was first mentioned in Korein and Badler [19]. Even though the shoulder and the wrist are firmly planted, we can still afford to move the elbow along a circle with its axis through the shoulder and the wrist (See Figure 3). The human figure has four limbs, two from arms and two from legs. The redundant DOF for the i-th limb linkage can be parameterized with a rotation angle θ_i, $1 \leq i \leq 4$, about the axis.

Without loss of generality, we assume that the positions and orientations of hands and feet are fixed by constraints. If there is a free hand or foot, the DOFs in the corresponding limb

are left unchanged. Let $\mathbf{m} = (\mathbf{p}, \mathbf{q}^1, \cdots, \mathbf{q}^r, \mathbf{q}^{r+1}, \cdots, \mathbf{q}^n)$ be the configuration of a human-like figure. Its rear part $(\mathbf{q}^{r+1}, \cdots, \mathbf{q}^n)$ denotes the DOFs for the limbs and the fore part $(\mathbf{p}, \mathbf{q}^1, \cdots, \mathbf{q}^r)$ does the remaining DOFs. Since the constraints restrain the DOFs in the limb linkages, the reduced set of parameters $(\mathbf{p}, \mathbf{q}^1, \cdots, \mathbf{q}^r, \theta_1, \cdots, \theta_4)$ span all possible configurations of the figure under the constraints.

Incorporating the idea of reduced-coordinate formulation into the numerical optimization framework, we can solve an inverse kinematics problem using a fewer number of optimization parameters $\hat{\mathbf{x}} = (x_0, \cdots, x_{3r+2}, \theta_1, \cdots, \theta_4) \in \mathbb{R}^{3r+7}$. Note that we have replaced the rear part of $\mathbf{x}$ with the elbow circle parameters $\theta_1, \cdots, \theta_4$ for limb linkages. Whenever we evaluate the objective function with new parameters $\hat{\mathbf{x}}$, the parameters $(\mathbf{p}, \mathbf{q}^1, \cdots, \mathbf{q}^r)$ are computed first by Equation (5), and then the others for $(\mathbf{q}^{r+1}, \cdots, \mathbf{q}^n)$ are uniquely determined by an analytical solver which takes $(\mathbf{p}, \mathbf{q}^1, \cdots, \mathbf{q}^r)$ and θ_i, $1 \le i \le 4$, as input. Then, we extract the unknown part $(x_{3r+3}, \cdots, x_{3n+2})$ of $\mathbf{x}$ from $(\mathbf{q}^{r+1}, \cdots, \mathbf{q}^n)$ to evaluate the objective function in Equation (6). The reduced-coordinate formulation uses a fewer number of parameters to yield faster convergence and fewer iterations to enhance the overall performance.



(a) Initial configuration (b) Elbow rotation

(d) Redundancy (c) Shoulder & Wrist rotation

Figure 4: The process for adjusting the arm posture

5.3 Arm and Leg Postures

Consider a limb linkage, for example, an arm linkage. Starting from an initial configuration, we sequentially adjust the joint angles for the elbow, the shoulder and the wrist of the arm linkage to place the hand at the desired position and orientation. We assume that the torso and the shoulder positions are given. Let l_1, l_2, r_1, r_2 and L be defined as follows (See Figure 4(a)):

l_1 = the length of the upper arm,
l_2 = the length of the fore arm,
r_1 = the distance from the elbow rotation axis to the shoulder,
r_2 = the distance from the elbow rotation axis to the wrist, and
L = the distance between the shoulder and the wrist.

To place the wrist at a position distant from the shoulder by L (See Figure 4(b)), the angle ϕ between upper and lower arms is given by

$$\phi = \cos^{-1}\left(\frac{l_1^2 + l_2^2 + 2\sqrt{l_1^2 - r_1^2}\sqrt{l_2^2 - r_2^2} - L^2}{2r_1 r_2}\right), \quad (7)$$

as illustrated in the appendix. Then, we bring the wrist to the goal position by adjusting the shoulder angles (See Figure 4(c)). In the subsequent step, we rotate the wrist angles to coincide with the goal orientation. Once one feasible solution is given, the other solutions can be obtained by rotating the elbow about the axis that passes through the shoulder and the wrist positions. Given θ_i, we can determine the arm posture uniquely (See Figure 4(d)). Similarly, we can determine a leg posture.

If L is longer than the arm length, $l_1 + l_2$, the elbow stretches as far as possible. On the other hand, if L is too small, then the elbow angle could violate its lower limit and thus is pulled back into the allowable range. In both cases, we cannot place the wrist at the exact position and thus the corresponding penalty function yields a positive value for the given torso configuration.

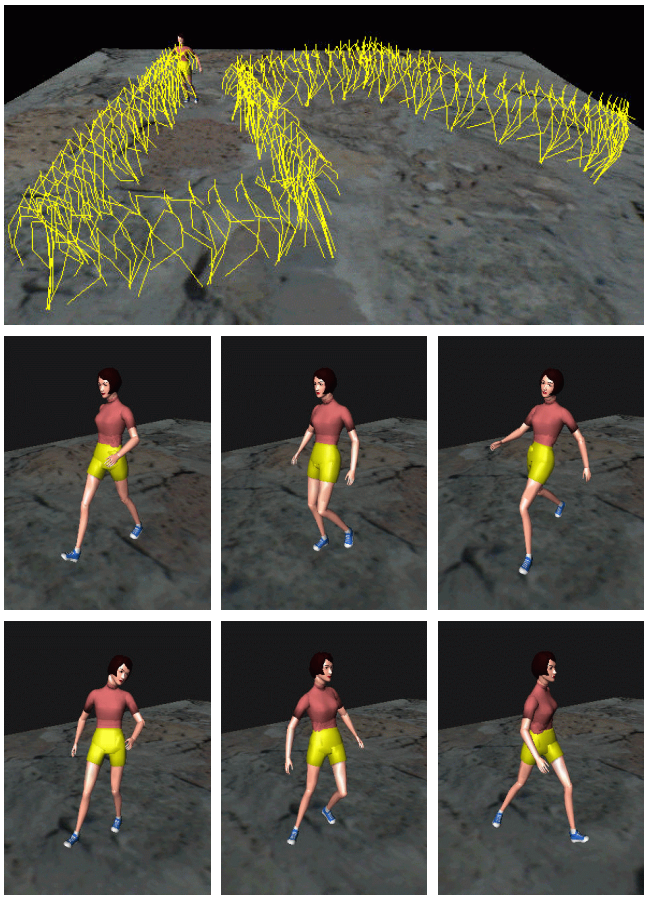
6 Experimental Results

Our human model has 6 DOFs for the pelvis position and orientation, 3 DOFs for the (either rigid or flexible) spine, and 7 DOFs for each limb to yield the total of 37 parameters for the inverse kinematics problem. Other parameters for the head, the neck, and the fingers are not used for resolving kinematic constraints. Through the reduced-coordinate formulation, we can remove 6 DOFs for each limb and thus we have at most 13 DOFs to be computed by a numerical optimization method. The motion clips for our experiments have been sampled at the rate of 30 frames per second.

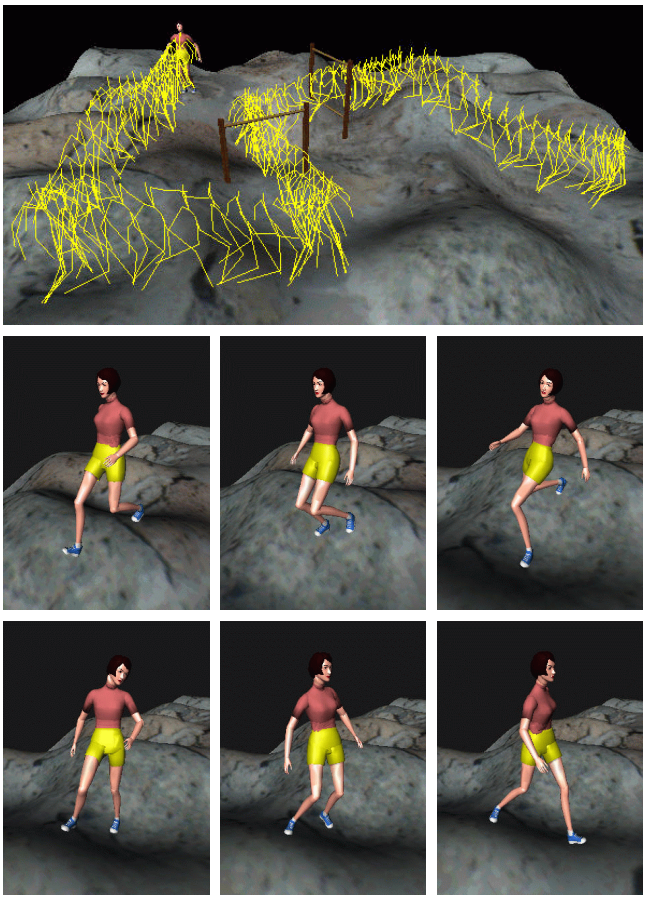
The walking motion of Figure 5(a) is produced by performing a sequence of transitions among a set of motion-captured clips, which include "walk straight", "turn left", "turn right", "start", and "stop". We interactively specify the moments of heel-strikes and toe-offs for the motion clips. This information is used for establishing the kinematic constraints that enforce the foot contacts for the entire motion. The terrain of Figure 5(b) is represented as a NURBS surface of which control points are placed on a regular grid with a spacing of 80 % of the height of the character, and their y-coordinates (heights) are randomly perturbed within 120 % of the height. To adapt the motion onto the rough terrain with doorways, we first adjust the constraints such that the contact positions are shifted along the y-axis to be placed on the terrain, and add new constraints to bend the character under the doorways. Then, we use our motion fitting algorithm to warp the motion to satisfy the constraints. The original and the adapted motions are depicted in Figures 5(a) and 5(b), respectively.

The "climbing a rope" example in Figure 5(c) gives constraints on both hands and feet. A physically simulated rope is used to explicitly illustrate the moments of grasping and releasing the rope by a hand, which correspond to the initiation and termination, respectively, of a variational constraint for that hand. We adapt this motion to a different character with longer legs and a shorter body and arms. For the character morphing example shown in Figure 5(d), the size of a character smoothly changes to have extremely long legs and a short body, and then to have extremely short legs and a long body. The original walking motion is warped to preserve its uniform stride against the change of character size.

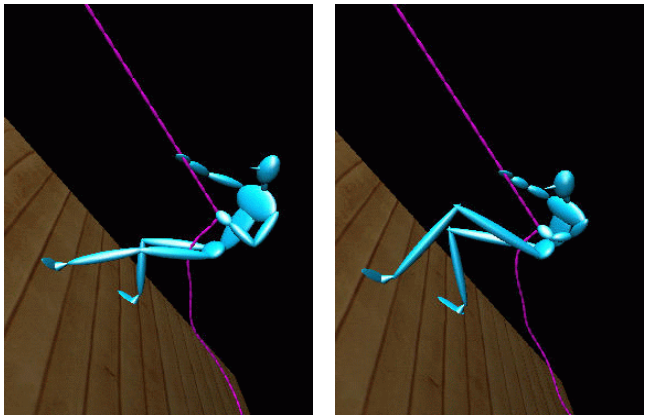
Our motion fitting method is also useful for generating a smooth transition between motion clips. Figure 5(e) shows the transitions from walking to sneaking and from sneaking to walking. The basic approach is very similar to the one presented by Rose *et al.* [27] We seamlessly connect the motion data by fading one out while fading the other in. Over the fading duration, Hermite interpolation and time warping techniques are used to smoothly blend the joint parameters of the motion data. Since joint parameter blending may cause foot sliding, we enforce foot contact constraints with the motion fitting method.



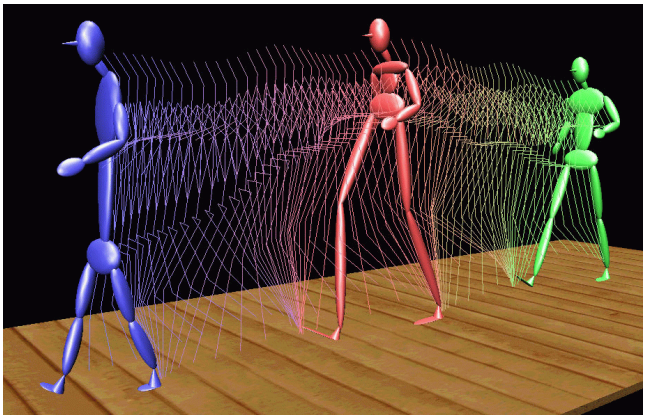
(a) The original walking motion on the flat ground



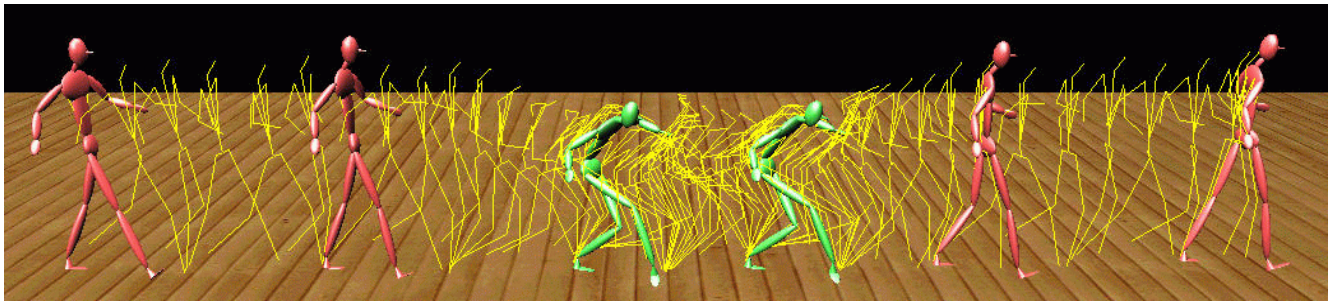
(b) The adapted motion for the rough terrain



(c) Climbing a rope for different characters



(d) Character morphing



(e) Transitions between walking and sneaking

Figure 5: Examples

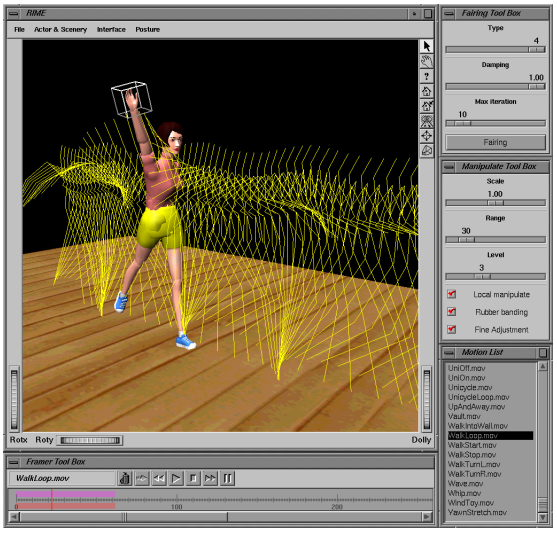


Figure 6: Graphical user interface

Table 1 gives a performance summary of the examples. Timing information is obtained on a SGI Indigo2 workstation with an R10000 195 MHz processor. The execution time for each example is not only influenced by quantitative factors such as the number of frames, constraints and parameters, but also by qualitative factors such as the difficulty of achieving desired features specified by constraints and the quality of initial estimates. In particular, well-chosen initial estimates provide great speedups for most of examples. One promising observation is that both execution times and maximum errors rapidly decrease level by level. This implies that the performance of our algorithm is not critically dependent on the error tolerance. In all examples, every constraint is satisfied within or slightly over 1 % of the height of the character by the hierarchical fitting of four levels. A few more levels may result in a more accurate solution. As shown in experimental data, we can anticipate that the computation cost for an additional level is much cheaper than the cost for the prior level.

Our prototype system is implemented in C++ on top of X-windows/MOTIF^{TM} and Open Inventor^{TM} that facilitates developing interactive 3D user interfaces. In particular, Open Inventor^{TM} provides convenient toolkits to support direct manipulation in the 3D space. With the toolkits, an animator can interactively modify the pose of a character by dragging one of its segments. Our user interface allows the user to edit a desired portion of the motion by adjusting the spacing of knots over which the displacement map is defined. Since we use cubic B-splines to represent the displacement map, the change of the pose at a frame could affect the poses at the neighboring frames covered by seven knots at the current level in the hierarchy [1, 7].

7 Discussion

In this section, we compare our motion fitting algorithm to the previous approaches at several viewpoints.

Interpolating splines vs. Multilevel B-splines: An interpolating spline is a possible choice to represent the displacement map [3, 37]. However, the interpolating spline could undulate rapidly for a dense distribution of constraints so that it often fails to preserve the detail characteristics of the original motion. Instead, we use a series of uniform B-splines that form a hierarchy of motion displacements. Since uniform B-splines approximate different details of displacements according to knot spacings, we are able to edit the motion at any level of detail; fitting at the coarsest level makes a gross deformation and then fine details are incrementally added at the finer levels.

Global vs. Local least-squares: Fitting a B-spline curve to scattered data points can be formulated as a least-squares approximation problem for solving an over-determined or under-determined system of linear equations [26]. To obtain the optimal solution in the least-squares sense, we can use the pseudo inverse of a large matrix that accommodates the data points all together [15]. However, this global method often suffers from over-shooting that may give an undesirable curve shape. Ironically, the less accurate local method in Equation (2) is preferred in a hierarchical framework. Since approximation errors at one level are incrementally canceled out in the later levels, the accuracy at each level is not critical. This local method is much more efficient and less prone to over-shooting than the global method.

Single large optimization vs. Many small optimizations: Gleicher [13] cast motion retargetting as a large optimization problem. Based on the "divide-and-conquer" strategy, however, we partition his optimization problem into many smaller inverse kinematics optimizations and then solve each of them very efficiently by adopting specialized techniques such as the hybrid inverse kinematics solver. Finally, we combine the poses at constrained frames employing the hierarchical curve fitting technique. This divide-and-conquer approach provides a noticeable speedup to satisfy the performance requirement of our interactive motion editing system.

Limitation of our approach: Our motion fitting algorithm requires additional work to handle an inter-frame constraint that enforces the relationship among parameters scattered over multiple frames. This kind of constraints are used for avoiding foot sliding between the heel-strike of a foot and its toe-off while allowing the absolute coordinates of the foot to be altered. We address this problem by incorpolating the intra-frame inverse kinematics constraints at those co-related frames into a larger optimization problem that includes inter-frame constraints among those frames as well. In the extreme case where all frames are related to each other, this approach is reduced to Gleicher's so that we have to solve one large optimization problem to derive the displacement map at each level in the hierarchy.

8 Conclusion

We have presented a new approach to adapting existing motion of a human-like character to have desired features that are specified by a set of constraints. The key idea of our approach is to introduce a hierarchical motion representation by which we cannot only manipulate a motion adaptively to satisfy a large set of constraints within a specified error tolerance, but also edit an arbitrary portion of the motion through direct manipulation.

The performance of our algorithm is greatly improved by employing a curve fitting technique that minimizes a local approximation error. The hierarchical structure compensates for the possible drawbacks of the local approximation method by globally propagating the displacement at a coarse level and later tuning at fine levels. Further performance gain is achieved by the new inverse kinematics solver. Our hybrid algorithm performs much faster than a pure numerical algorithm. We also present a simple yet effective formulation to optimize orientation parameters without yielding extra constraints.

Table 1: Performance data. # of parameters counts the DOFs of a character used for resolving kinematic constraints. A number in ( ) indicates that of reduced parameters. The maximum error is measured in percentages of the height of the character.

	walking on rough terrain				climbing a rope			
# of parameters	20 (8)				37 (13)			
# of frames	464				275			
# of constraints (except joint limits)	5568				5898			
preprocessing time (CPU sec.)	0.17				0.21			
level	1st	2nd	3rd	4th	1st	2nd	3rd	4th
# of frames/knot	16	8	4	2	16	8	4	2
execution time (CPU sec.)	2.50	1.67	1.31	1.10	6.50	3.51	1.32	0.70
maximum error (%)	14.31	9.24	3.48	1.06	11.09	7.51	3.57	1.11

	character morphing				transition			
# of parameters	20 (8)				20 (8)			
# of frames	62				119			
# of constraints (except joint limits)	420				834			
preprocessing time (CPU sec.)	0.02				.			
level	1st	2nd	3rd	4th	1st	2nd	3rd	4th
# of frames/knot	16	8	4	2	16	8	4	2
execution time (CPU sec.)	0.18	0.13	0.09	0.06	0.18	0.15	0.15	0.14
maximum error (%)	6.98	5.35	3.08	1.10	7.17	4.71	2.14	0.74

Acknowledgments

We would like to thank Michael Cohen at Microsoft Research for his careful reading and constructive criticism. We are also grateful to Hyun Joon Shin and Min Gyu Choi who helped edit the accompanying video tape. We thank Korean Broadcasting System (KBS) for supplying the model in Figures 5(a) and 5(b). This research was supported in part by the Ministry of Information and Communication (MIC) and the Electronics and Telecommunications Research Institute (ETRI).

References

[1] R. H. Bartels, J. C. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1979.

[2] L. S. Brotman and A. N. Netravali. Motion interpolation by optimal control. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):309–315, August 1988.

[3] A. Bruderlin and L. Williams. Motion signal processing. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 29:97–104, August 1995.

[4] A. Certain, J. Popović, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. *Computer Graphics (Proceedings of SIGGRAPH 96)*, 30:97–115, August 1996.

[5] M. F. Cohen. Interactive spacetime control for animation. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):293–302, July 1992.

[6] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 29:173–182, August 1995.

[7] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design – A Practical Guide*. Academic Press, 4th edition, 1997.

[8] A. Finkelstein and D. H. Salesin. Multiresolution curves. *Computer Graphics (Proceedings of SIGGRAPH 94)*, 28:261–268, July 1994.

[9] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):205–212, August 1988.

[10] D. R. Forsey and R. H. Bartels. Surface fitting with hierarchical splines. *ACM Transactions of Graphics*, 14(2):134–161, April 1995.

[11] M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged figures. *Computer Graphics (Proceedings of SIGGRAPH 85)*, pages 263–270, July 1985.

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

[13] M. Gleicher. Retargetting motion to new characters. *Computer Graphics (Proceedings of SIGGRAPH 98)*, 32:33–42, July 1998.

[14] S. Guo and J. Robergé. A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Proceedings of Computer Animation and Simulation '96, Eurographics Animation Workshop*, pages 95–107. Springer-Verlag, 1996.

[15] W. M. Hsu, J. F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26:177–184, July 1992.

[16] W. Kahan. Lectures on computational aspects of geometry. *Unpublished manuscripts*, 1983.

[17] M. J. Kim, M. S. Kim, and S. Y. Shin. A general construction scheme for unit quaternion curves with simple high order derivatives. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 29:369–376, August 1995.

[18] Y. Koga, K. Kondo, J. Kuffer, and J. Latombe. Planning motions with intentions. *Computer Graphics (Proceedings of SIGGRAPH 94)*, 28:395–408, July 1994.

[19] J. U. Korein and N. I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE CG&A*, pages 71–81, Nov. 1982.

[20] S. Lee, K.-Y. Chwa, S. Y. Shin, and G. Wolberg. Image metamorphosis using snakes and free-form deformations. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 29:439–448, August 1995.

[21] S. Lee, G. Wolberg, and S. Y. Shin. Scattered data interpolation with multilevel b-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):228–244, 1997.

[22] Z. Liu, S. G. Gortler, and M. F. Cohen. Hierarchical spacetime control. *Computer Graphics (Proceedings of SIGGRAPH 94)*, pages 35–42, July 1994.

[23] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, 1997.

[24] B. Paden. *Kinematics and Control Robot Manipulators*. PhD thesis, University of California, Berkeley, 1986.

[25] J. C. Platt and A. H. Barr. Constraint methods for flexible models. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):279–288, August 1988.

[26] W. H. Press, Saul A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[27] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and Adverbs: Multidimensional motion interpolation. *IEEE CG&A*, 18(5):32–40, October 1998.

[28] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. *Computer Graphics (Proceedings of SIGGRAPH 96)*, 30:147–154, August 1996.

[29] F. J. M. Schmitt, B. A. Barsky, and W. Du. An adaptive subdivision method for surface-fitting from sampled data. *Computer Graphics (Proceedings of SIGGRAPH 86)*, 20(4):179–188, August 1986.

[30] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 29:161–172, August 1995.

[31] K. Shoemake. Animating rotation with quaternion curves. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19:245–254, 1985.

[32] D. Tolani and N. I. Badler. Real-time inverse kinematics of the human arm. *Presence*, 5(4):393–401, 1996.

[33] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 29:91–96, August 1995.

[34] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2):157–166, July 1992.

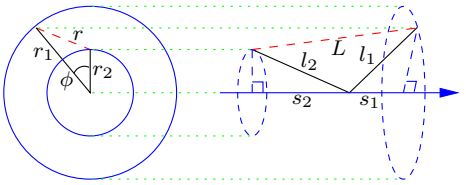
[35] D. J. Wiley and J. K. Hahn. Interpolation synthesis for articulated figure motion. In *Proceedings of IEEE Virtual Reality Annual International Symposium '97*, pages 157–160. IEEE Computer Society Press, 1997.

[36] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4):159–168, August 1988.

[37] A. Witkin and Z. Popović. Motion warping. *Computer Graphics (Proceedings of SIGGRAPH 95)*, 29:105–108, August 1995.

[38] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.

Appendix: Derivation for Equation (7)



To identify the angle between the upper and fore arms, we project the joint positions onto a plane perpendicular to the elbow rotation axis. Then, the projections for the shoulder and the wrist are placed on two concentric circles whose center coincides with the projection for the elbow. The distance r between the projections is given in terms of r_1, r_2 and the angle ϕ between them.

$$r^2 = r_1^2 + r_2^2 - 2r_1 r_2 \cos\phi.$$

Letting $s_1 = \sqrt{l_1^2 - r_1^2}$ and $s_2 = \sqrt{l_2^2 - r_2^2}$, respectively, the distance L between the shoulder and the wrist positions is

$$\begin{aligned} L^2 &= (s_1 + s_2)^2 + r^2 \\ &= l_1^2 + l_2^2 + 2\sqrt{l_1^2 - r_1^2}\sqrt{l_2^2 - r_2^2} - r_1^2 - r_2^2 + r^2 \\ &= l_1^2 + l_2^2 + 2\sqrt{l_1^2 - r_1^2}\sqrt{l_2^2 - r_2^2} - 2r_1 r_2 \cos(\angle r_1 r_2). \end{aligned}$$

Therefore, $\cos\phi = \frac{l_1^2 + l_2^2 + 2\sqrt{l_1^2 - r_1^2}\sqrt{l_2^2 - r_2^2} - L^2}{2r_1 r_2}$.