

Hierarchical Spacetime Control

Zicheng Liu
zl@cs.princeton.edu

Steven J. Gortler
sjg@cs.princeton.edu

Michael F. Cohen
mfc@cs.princeton.edu

Department of Computer Science
Princeton University

Abstract

Specifying the motion of an animated linked figure such that it achieves given tasks (e.g., throwing a ball into a basket) and performs the tasks in a realistic fashion (e.g., gracefully, and following physical laws such as gravity) has been an elusive goal for computer animators. The *spacetime constraints* paradigm has been shown to be a valuable approach to this problem, but it suffers from computational complexity growth as creatures and tasks approach those one would like to animate. The complexity is shown to be, in part, due to the choice of finite basis with which to represent the trajectories of the generalized degrees of freedom. This paper describes new features to the spacetime constraints paradigm to address this problem.

The functions through time of the generalized degrees of freedom are reformulated in a hierarchical wavelet representation. This provides a means to automatically add detailed motion only where it is required, thus minimizing the number of discrete variables. In addition the wavelet basis is shown to lead to better conditioned systems of equations and thus faster convergence.

CR Categories and Subject Descriptors: I.3.7 [Computer-Graphics]: *Three Dimensional Graphics and Realism* ; I.6.3 [Simulation and Modeling]: *Applications* ; G.1.6 [Constrained Optimization]

Additional Key Words and Phrases: Animation, Spacetime, Wavelets.

1 Introduction

The spacetime constraint method, proposed in 1988 by Witkin and Kass [18], and extended by Cohen [5], has been shown to be a useful technique for creating physically based and goal directed motion of linked figures. The basic idea of this approach can be illustrated with a three-link arm and a ball (see Figure 1). The problem statement begins with specifying *constraints*, examples being specifying the position of the arm at a given time, requiring the ball to be in the hand (end effector) at time t_0 , and that the arm is to throw the ball at time t_1 to land in a basket at time t_2 . In addition, the animator must specify an *objective* function, such as to perform the tasks specified by the constraints with minimum energy or some other style consideration. The solution to such a series of specifications is a set of functions through time (or *trajectories*) of each degree of freedom (DOF), which in this case are the joint angles of the arm. Thus the unknowns span both space (the joint angles) and time, and have led to the term *spacetime constraints*.

Related approaches to the spacetime constraint paradigm are reported in [17, 12]. In each of these papers, feedback control

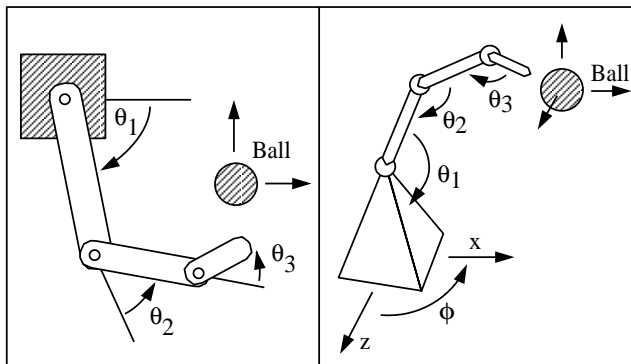


Figure 1: A planar three-link arm and a 6 DOF basketball player.

strategies are the fundamental unknown functions rather than DOF trajectories. The goal is set, for example, for the creature to move in some direction as far as possible in 10 seconds, and a *score* for a particular motion is defined as the distance traveled. An initial control strategy is selected, a dynamic simulation is run and the results are scored. Iterations change the control strategy, as opposed the motion curves, producing a simulation that, hopefully, has a higher score. The results of these studies are encouraging, however, they are distinctly different from that in the previous spacetime constraint work (and the work described in this paper) in which the aim is to provide the animator with the overall control of the motion.

The spacetime constraint formulation leads to a non-linear constrained variational problem, that in general, has no closed form solution. In practice, the solution is carried out by reducing the space of possible trajectories to those representable by a linear combination of basis functions such as cubic B-splines. Finding the finite number of coefficients for the B-splines involves solving the related constrained optimization problem, (i.e., finding the coefficients to create motion curves for the DOF that minimize the objective while satisfying the constraints). Unfortunately, general solutions to such a non-linear optimization problem are also unknown.

Based on this observation, Cohen developed an interactive spacetime control system using hybrid symbolic and numeric processing techniques [5]. In this system, the user can interact with the iterative numerical optimization and can *guide* the optimization process to converge to an acceptable solution. One can also focus attention on subsets or *windows* in spacetime. This system produces physically based and goal directed motions, but it still suffers from a number of computational difficulties, most notably as the complexity of the creature or animation increases. Addressing this problem is the central focus of this paper.

One problem that arises is the symbolic processing of the constraints and objective, and the subsequent evaluation of expressions. Constraints and objectives are entered into the system by the user and/or by automatic construction of the equations of motion from the linkage description. These may be any second order

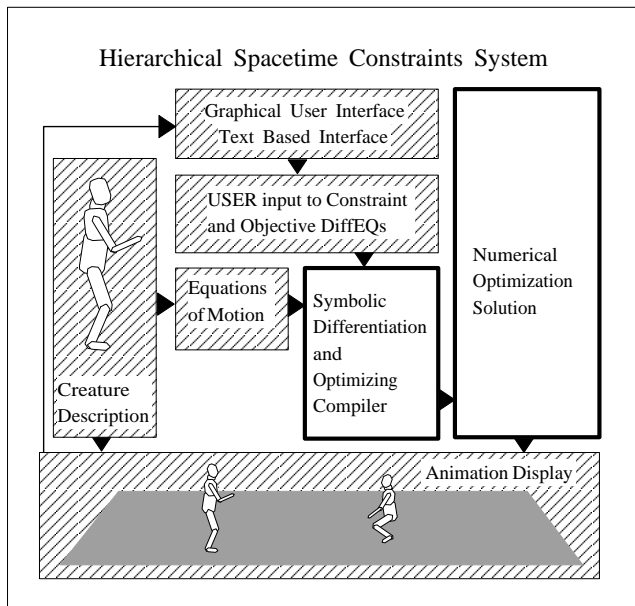


Figure 2: The Hierarchical Spacetime Constraints System. This paper focuses on the Symbolic Differentiation and Optimizing Equation Compiler, and the Numerical Optimization System

differential equation of the linkage DOF. The symbolic processing stage is responsible for deriving the first (and possibly second) derivatives of each constraint expression and the objective with respect to each DOF. The result is the construction of evaluation trees to evaluate each expression and its derivatives. Unfortunately, even a planar three-link arm can result in evaluation trees containing over 145,000 nodes, that must be evaluated multiple times during the optimization process. The process described above is closely related to that of compiler design and optimization [1]. In particular, the compiler optimization technique of *common subexpression elimination* has been shown to be of great value in reducing the size of the evaluation trees and thus greatly speeds up the symbolic computation of the derivative expressions and the subsequent numerical evaluations [11]. In our experiments, common subexpression elimination resulted in one to two orders of magnitude of reduction in evaluation of the resulting expressions.

A more important difficulty in the spacetime system is that the user is required to choose the discretization of the B-spline curves. If not enough control points are selected, there may be no feasible solution (i.e., one that meets all constraints), or the restriction to the curve is so severe, that the resulting motion curves have a much higher objective cost than necessary. If too many control points are selected, then the computational complexity is increased unnecessarily due to the larger number of unknowns as well as the resulting ill-conditioning of the linear subproblems that arise in the solution [16]. This complexity issue is addressed by reformulating the DOF functions in a *hierarchical* basis, in particular, in a B-spline wavelet (B-wavelet) basis. Wavelets provide a natural and elegant means to include the proper amount of local detail in regions of spacetime that require the extra subdivision without overburdening the computation as a whole.

2 System overview

The interactive spacetime control system is shown in Figure 2. Input to the system includes user defined constraints and objectives and a creature description from which the symbolic equations of motion are generated automatically. The equations of motion

define the torque at each joint as a function of the position and velocity of all joints as well as physical properties such as mass and length of the links. These expressions for torque are central to the definition of a minimum energy objective. The expressions are next symbolically differentiated and compiled to create concise evaluation trees.

The main focus of this paper is on the next section, the numerical process that solves for the coefficients of the chosen B-spline or hierarchical wavelet basis. Finally, the intermediate and final animations are displayed graphically. The animator can simply watch the progress of the optimization procedure or can interact directly with the optimization by creating starting motion curves for the DOF and/or by modifying intermediate solutions.

2.1 Symbolic Constraints and Objectives

An important feature of the interactive spacetime constraints system is the ability to specify and modify constraints and the objective at run-time. This requires a high level graphical and/or textual interface to communicate the animator's intentions. The constraints and/or objective may be any second order integro-differential expression. A simple language interface has been developed for this specification with a syntax much like general mathematical expressions. In most cases, a constraint is a simple differential expression such as that a particular joint, at a particular point in time must have a given value, or that its velocity must be zero or some other value. The expressions arising from the equations of motion are much longer and include many transcendental functions. In addition, these expressions form the terms of the integrand of the minimum energy objective defined by integrating the square of the joint torques over time. For example, for the three link arm

$$f(\Theta) = \int_{t_0}^{t_f} \tau_1^2 + \tau_2^2 + \tau_3^2 dt \quad (1)$$

where Θ are the generalized DOF (i.e., the joint angles), and τ_i is the torque about joint i . Numerical quadrature of such expressions can then be carried out by evaluating the expression at multiple values of T .

2.2 Expression Differentiation and Compilation

The symbolic expressions for constraints and objectives are then *compiled* for evaluation during the numerical optimization process. Compiler designers are faced with similar problems and thus many techniques from the compiler optimization literature [1] are applicable here. Related work is also found in the literature on automatic differentiation [15], and similar work to the current application is described in the CONDOR system by Kass [11].

The compiler developed in the spacetime problem begins with a bottom up parser that produces an evaluation tree for each expression. The expression can then be evaluated by inserting the DOF values in the leaves of the tree and recursively evaluating nodes upward until the value of the expression is contained at the topmost node. The expressions arising in this context, however, often contain many common subexpressions. The problems in building and evaluating the expressions is exacerbated by the fact that the optimization routines require gradients (Jacobians) and possibly Hessians of the constraints and objective.

To avoid extra evaluation, common subexpressions are extracted by recursively moving from the leaves to the top node, making a list of unique nodes and checking each new node against the list. In the case of leaf nodes that are always variable ID's, this is simple, for internal function nodes their children must be checked, and in the case of commutative operations both orderings must be checked.

As an example of the power of the common subexpression elimination (CSE), the expression trees for the three link arm without CSE contained 145,584 nodes, compared to 2,932 nodes after CSE¹!

Once the reduced expression trees are built, each expression is evaluated one or more times per optimization iteration. The greatest savings occur in the evaluation of the integrals that comprise the objective since the numerical quadratures request multiple evaluations of the compiled expressions with different values of T . The change in T directly affects only the values inserted at the leaves of the trees, and thus a single compilation may be used hundreds or thousands of times.

3 Hierarchical B-splines

In the most general setting, the trajectory of a DOF, $\theta(t)$, can be any function in L^2 . In practice, however, solving the spacetime constraint problem requires restricting the solution to some finite dimensional function space, leading to a finite number of scalar unknowns. The possible trajectories a particular DOF can take are thus restricted to be a linear combination of basis functions chosen to represent the DOF motion curve. In other words, given n basis functions,

$$\theta(t) = \sum_{i=1}^n c_i \phi_i(t) \quad (2)$$

where the coefficients c_i scale the basis functions $\phi_i(t)$. In the spacetime constraint systems to date, Witkin and Kass [18] used discretized functions consisting of evenly spaced points in time from which derivatives were approximated by finite differencing. Cohen represented the DOF functions as uniform cubic B-spline curves, with some provision to change the resolution of the B-splines within specified regions of the curve.

The more basis functions and corresponding coefficients that are used, the larger the space of possible solutions. Unfortunately, for two reasons, one pays a high cost in terms of computational resources for this extra freedom. The extra unknown coefficients translate into larger subproblems at each iteration of the solution. In addition, discretizations of this type also lead to ill-conditioned systems requiring more iterations to solve [16].

Ideally, one would like to select a function space with just enough freedom to allow an almost optimal answer. In smooth portions of the trajectories, basis functions can be wider and in regions where the trajectory varies quickly, there should be more, narrower bases. Unfortunately, the optimal trajectories are not known in advance and thus a more flexible basis must be developed that can *adapt* to the local detail in the trajectories as the iterative solution proceeds.

Hierarchical systems of basis functions offer just this type of adaptivity. Hierarchical B-splines have been used in the context of shape design [8] to allow modification of curves and surfaces at levels of detail selected by the user. The hierarchical B-spline basis consists of a pyramid of translations and dilations of B-splines (the rows labeled V in Figure 3) ranging from very wide B-splines at the top to finer scaled basis functions below.

Each level going down contains twice as many basis functions per unit length. This hierarchical basis has attractive properties for use in describing the trajectories in the current application. However, this is a *redundant* basis, since any function realizable at one level can also be created from the finer basis functions below. In addition, how to achieve the desired adaptivity is not immediately apparent.

¹It should be noted that further reductions in the DAGs could be made through the use of trigonometric identities, however, this has not been done in the current implementation.

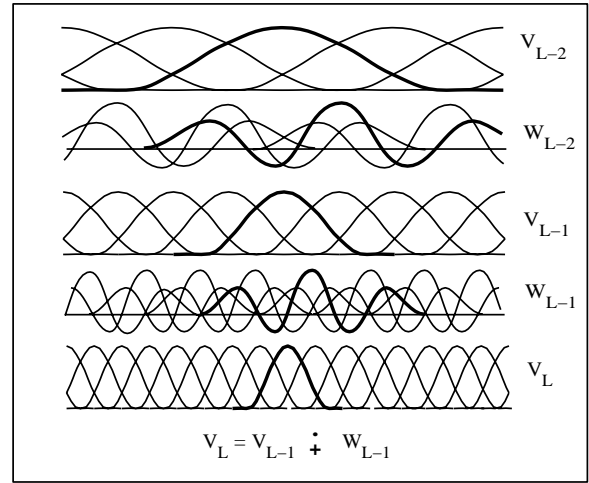


Figure 3: Hierarchy of B-spline and Wavelet Bases.

4 Wavelets

A more elegant and concise hierarchical basis, and one that leads naturally to an adaptive basis, is offered by a *wavelet* construction. This section concentrates on the advantages of wavelets and wavelet formulations in the spacetime animation problem.

The primary difference between wavelets and hierarchical B-splines is that the wavelet coefficients at each level represent *differences* from the levels above as opposed to directly representing the local function value. Also, unlike hierarchical B-splines, each new layer is not redundant with those above but rather adds only *local detail* in the result at some resolution.

4.1 Advantages of Wavelets to Spacetime Animation

The wavelet construction results in a non-redundant basis that provides the means to begin with a low resolution basis and then *adaptively refine* the representation layer by layer when necessary without changing the representation above. If refinements are required in only part of the interval, then only those coefficients whose bases have support in this region need to be added.

Since the wavelet coefficients encode differences, in smooth portions of the trajectory the coefficients encoding finer scale detail will be zero. Thus, only those basis functions with resulting coefficients greater than some ϵ will have a significant influence on the curve and the rest can be ignored. In other words, given an *oracle* function [10, 9], (discussed later) that can predict which coefficients will be above a threshold, only the corresponding subset of wavelets needs to be included.

Solutions to the non-linear spacetime problem, as discussed in more detail below, involve a series of quadratic subproblems for which the computational complexity depends on the number of unknown coefficients. The smaller number of significant unknown coefficients in the wavelet basis provide faster iterations. In addition, the wavelet basis provides a better conditioned system of equations than the uniform B-spline basis, and thus requires less iterations. The intuition for this lies in the fact that there is no single basis in the original B-spline basis that provides a global estimate of the final trajectory (i.e., the locality of the B-spline basis is, in this case, a detriment). Thus, if the constraints and objective do not cause interactions across points in time, then information about changes in one coefficient travels very slowly (in $O(n)$ iterations) to other parts of the trajectory. In contrast, the hierarchical wavelet basis provides a shorter ($O(\log(n))$) “communication” distance between any two basis functions. This is the

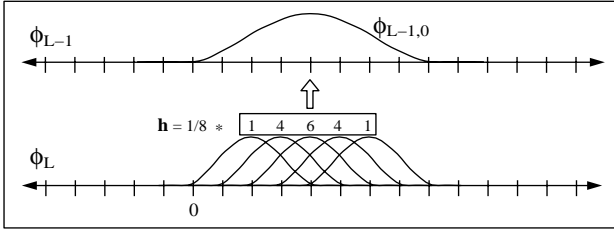


Figure 4: Five B-splines $\phi_{L,j}$ may be combined using the weights h to construct the double width B-spline $\phi_{L-1,0}$

basic insight leading to *multigrid* methods [16], and the related hierarchical methods discussed here.

An additional benefit involves the integration of the objective. Since a lower resolution results, by definition, in a smoother trajectory, less samples must be taken during the numerical quadrature. As wavelets are added in particular regions, the sampling density only needs to be increased in these areas.

Finally, the wavelet representation allows the user to easily lock in the coarser level solution and only work on details simply by removing the coarser level basis functions from the optimization. This provides the means to create small systems that solve very rapidly to develop the finest details in the trajectories.

4.2 Wavelet Construction: Two-Part Basis

To explain the complete wavelet construction, a first step will be to create a new basis for the same space of functions defined by the B-splines, but consisting of two distinct types of basis functions. To understand the two-part basis, begin with the familiar cubic B-spline basis made up of translated copies of the B-spline blending function $\phi(t)$. Let us denote the basis functions as

$$\phi_{L,j}(t) = \phi(t - j) \quad (3)$$

(the index j represents the translation of a specific basis function from the canonical B-spline basis function left justified at zero [2], and L is the *level* or resolution of the basis). The space (or family) of functions spanned by all linear combinations of these basis functions will be denoted V_L . V_L contains all functions that are piecewise cubic between adjacent integers, and are C^2 (have simple knots at the integers).

Wavelets offer an alternative basis for the same space V_L , in particular, a hierarchical basis. But let us begin by building a two-part basis at level $L - 1$ of the hierarchy (i.e., at half the resolution). The two-part basis begins with the basis functions

$$\phi_{L-1,j}(t) = \phi(2^{-1}t - j) \quad (4)$$

These basis functions are twice as wide as the original B-spline basis functions, and hence the space they span contains piecewise cubic functions with simple knots at all *even* integers. This space will be referred to as V_{L-1} . According to the well known B-spline knot insertion algorithm [2, 4] there is the following relationship

$$\phi_{L-1,j} = \sum_k h_{k-2j} \phi_{L,k} \quad (5)$$

where the sequence h is given in the appendix. See Figure 4.

Clearly V_{L-1} is a proper subset of V_L and thus, it is not as rich as the space V_L . Therefore, to find a new basis for V_L , more basis functions are needed than just those that span V_{L-1} . In the wavelet methodology, this is accomplished by introducing into the basis, translated copies of a special wavelet shape ψ . Just as with the B-splines, the relationship between the wavelet basis functions and the model wavelet shape is notated $\psi_{L-1,j}(t) = \psi(2^{-1}t - j)$. These new basis functions are defined as:

$$\psi_{L-1,j} = \sum_k g_{k-2j} \phi_{L,k} \quad (6)$$

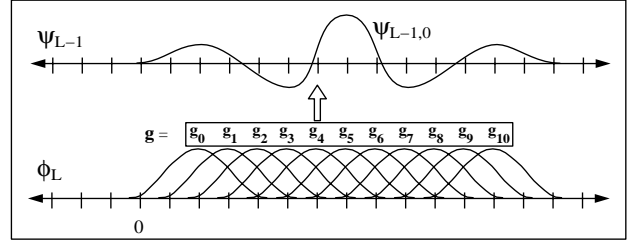


Figure 5: Eleven B-splines $\phi_{L,j}$ may be combined using the weights g to construct the wavelet function $\psi_{L-1,0}$

where the sequence g is given in the appendix. See Figure 5. There is some degree of freedom in choosing the sequence g , as long as the new basis functions “fill in” the missing space between V_L and V_{L-1} . In other words the $\psi_{L-1,j}$ must span a function space W_{L-1} such that $V_{L-1} \dot{+} W_{L-1} = V_L$ (direct sum). See Figure 3.

The sequence used here was chosen so that the basis functions $\psi_{L-1,j}$ are orthogonal to the basis functions $\phi_{L-1,j}$ with respect to the L_2 inner product, defined for general functions f and g as

$$\langle f(t), g(t) \rangle = \int_{-\infty}^{+\infty} f(t)g(t) dt \quad (7)$$

This type of construction is called *semi-orthogonal*. (In an orthogonal construction *all* of the basis functions are orthogonal to each other.) Other advantages to this particular choice of g include compactness and symmetry.

With this construction, we now have two alternate bases for V_L , the B-spline basis

$$\{\phi_{L,j}\} \quad (8)$$

and the *two-part* basis

$$\{\phi_{L-1,j}, \psi_{L-1,j}\} \quad (9)$$

Just as the two-part basis functions can be expressed as a combination of the B-splines basis functions (Equations (5) and (6)), so too, the B-spline basis functions can theoretically be expressed as combinations of the two-part basis functions. This is given by

$$\phi_{L,j} = \sum_k \tilde{h}_{j-2k} \phi_{L-1,k} + \sum_k \tilde{g}_{j-2k} \psi_{L-1,k} \quad (10)$$

The sequences \tilde{h} and \tilde{g} which are described in [4] have infinite length but decay quickly from their centers. In the literature, there are many wavelet constructions, each with its own particular functions ϕ and ψ , and sequences h , \tilde{h} , and \tilde{g} , with varying degrees of orthogonality, compactness, and smoothness. The sequences and the particular wavelet construction described in this paper are derived in [4], and were chosen because of the semi-orthogonality of the basis, the associated ϕ is a cubic B-spline (i.e., C^2), and the wavelet function ψ is symmetric with compact support.

Suppose some DOF function $\theta(t)$ in V_L has been expressed as a linear combination of the B-spline basis functions

$$\theta(t) = \sum_j c_{\phi_{L,j}} \phi_{L,j} \quad (11)$$

where the c are scalar coefficients. The coefficients needed to express the function in the two-part basis can be found using the following formula

$$\begin{aligned} c_{\phi_{L-1,j}} &= \sum_k \tilde{h}_{k-2j} c_{\phi_{L,k}} \\ c_{\psi_{L-1,j}} &= \sum_k \tilde{g}_{k-2j} c_{\phi_{L,k}} \end{aligned} \quad (12)$$

and now

$$\theta(t) = \sum_j c_{\phi_{L-1,j}} \phi_{L-1,j}(t) + \sum_j c_{\psi_{L-1,j}} \psi_{L-1,j}(t) \quad (13)$$

(This process is similar to the one illustrated in Figures (4-5) except that h and g and are interchanged with \tilde{h} and \tilde{g}). Intuitively speaking, the $c_{\phi_{L-1,j}}$ encode the smooth (low frequency) information about the function $\theta(t)$, and the $c_{\psi_{L-1,j}}$ encode the detail (higher frequency) information. In a semi-orthogonal construction (as well as a fully orthogonal one), the smooth component $s(t) = \sum_j c_{\phi_{L-1,j}} \phi_{L-1,j}(t)$ is the orthogonal projection of $\theta(t)$ into V_{L-1} . Thus, it is the best approximation of $\theta(t)$ in the space V_{L-1} where the error is measured by

$$\| \epsilon(t) \| = \langle \epsilon, \epsilon \rangle^{1/2} \quad (14)$$

Alternatively if $\theta(t)$ has been represented with respect to the two-part basis, the representation with respect to the B-spline basis can be found with

$$c_{\phi_{L,j}} = \sum_k h_{j-2k} c_{\phi_{L-1,k}} + \sum_k g_{j-2k} c_{\psi_{L-1,k}} \quad (15)$$

4.3 Wavelets on the Interval

In a classical wavelet construction, the index j goes from $-\infty \dots \infty$, and V_L includes functions of unbounded support. In an animation context, only functions over some fixed finite interval of time need to be expressed, and it is important to only deal with a finite number of basis functions. Therefore, the space V_L used here will be the space of all C^2 functions defined over the interval $[0 \dots 2^L]$ that are piecewise cubic between adjacent integers (simple knots at the inner integers and quadruple knots at the boundaries). A basis for V_L is made up of *inner* basis functions, which are just those translational basis functions $\phi_{L,j}$ from Section 4.2 whose support lies completely within the interval, as well as three special *boundary* B-spline basis functions at each end of the interval. For the boundary basis functions, one may either choose to include the translational basis functions $\phi_{L,j}$ themselves from Section 4.2 whose support intersects the boundaries by just truncating those basis functions at the boundary, or else one may use the special boundary basis functions that arise from placing quadruple knots at the boundaries [2]. This complete set of basis functions will be denoted $\phi_{L,j}$ with j in $\{-3 \dots 2^L - 1\}$, where it is understood that the first and last three basis functions are the special boundary B-spline basis functions.

The two-part basis for V_L begins with the wider B-spline functions $\phi_{L-1,j}$ with j in $\{-3 \dots 2^{L-1} - 1\}$ where again the first and last three basis functions are scaled versions of the special boundary B-splines functions. The two-part basis is completed with the wavelet functions $\psi_{L-1,j}$ with j in $\{-3 \dots 2^{L-1} - 4\}$. Here too, the *inner* wavelet basis functions are just those translational functions $\psi_{L-1,j}$ from Section 4.2 that do not intersect the boundaries, while the first three and the last three interval wavelet basis functions must be specially designed to fit in the interval and still be orthogonal to the $\phi_{L-1,j}$. A full description of this construction is given in [3, 14].

This interval construction, which on the real line corresponds to Equations (5) and (6), is described by the linear time procedure **basis_xform_up** that is given in the appendix. As this procedure is equivalent to multiplication with a banded matrix, the inverse procedure **basis_xform_down** which describes the interval version of Equation (10) can be implemented by solving the banded linear system. This too can be done in linear time.

The interval version of Equation (15) can be implemented with the procedure **coef_xform_down**. And the inverse transformation, which is the interval equivalent of Equation (12) may be implemented by the procedure **coef_xform_up**.

4.4 Complete Wavelet Basis

The reasoning that was used to construct the two-part basis can now be applied recursively $L - 3$ times to construct a multilevel *wavelet basis*. Thus far, a two-part basis $\{\phi_{L-1,j}, \psi_{L-1,j}\}$ has been discussed as an alternative for the B-spline basis $\{\phi_{L,j}\}$.

Note that roughly half of the basis functions in the two-part basis are themselves B-spline basis functions (only twice as wide). To continue the wavelet construction, keep the basis functions $\psi_{L-1,j}$ and re-apply the reasoning of section 4.2 to replace the $\phi_{L-1,j}$ with $\{\phi_{L-2,j}, \psi_{L-2,j}\}$. This results in the new basis $\{\phi_{L-2,j}, \psi_{L-2,j}, \psi_{L-1,j}\}$.

Each time this reasoning is applied, the number of B-spline functions in the hierarchical basis is cut in half (roughly), and the new basis functions become twice as wide. After $L - 3$ applications², the wavelet basis

$$\{\phi_{3,k}, \psi_{i,j}\} \quad (16)$$

is obtained, with i in $\{3 \dots L - 1\}$, k in $\{-3 \dots 7\}$ and j in $\{-3 \dots 2^i - 4\}$, where the inner basis functions are defined by

$$\begin{aligned} \phi_{i,j}(t) &= \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= \psi(2^{(i-L)}t - j) \end{aligned} \quad (17)$$

This basis is made up of eleven wide B-splines, and translations (index j) and scales (index i) of the wavelet shape (as well as scales of the boundary wavelet basis functions).

The coefficients representing some function in the B-spline basis can be transformed to the full wavelet basis using the procedure **coef_pyrm_up**, that makes $L - 3$ calls to **coef_xform_up** each time with an input vector of $1/2$ the length. (Note since this transforms an n-vector to an n-vector, it can be implemented with proper indexing using linear storage).

```
coef_pyrm_up( b_in [], b_out [], w_out [] [], L )
  b_temp[L][] = b_in [] ;
  for( i = L; i >= 4; i -- )
    coef_xform_up( b_temp[i][], b_temp[i-1][],
                  w_out[i-1][], i ) ;
  b_out [] = b_temp[3][] ;
```

The inverse transformation is

```
coef_pyrm_down( b_in [], w_in [] [], b_out [], L )
  b_temp[3][] = b_in [] ;
  for( i = 4; i <= L; i ++ )
    coef_xform_down( b_temp[i-1][], w_in[i-1][],
                    b_temp[i][], i ) ;
  b_out [] = b_temp[L][] ;
```

Finally, the basis transformations **basis_pyrm_up** and **basis_pyrm_down** are identical to the above procedures, with the exception of replacing the procedures **coef_xform_up** and **coef_xform_down** with **basis_xform_up** and **basis_xform_down**.

The running time of these pyramid procedures is governed by the geometric series $(n + \frac{n}{2} + \frac{n}{4} + \dots + 1) = O(n)$, and hence they run in linear time. Each one of these four procedures transforms one n-vector, to another, and thus can be represented as a matrix. If M is the matrix of the linear transformation performed by the procedure **coef_pyrm_up**, then M^{-1} is the matrix of

²This process is stopped after $L - 3$ applications so that the three left boundary basis functions don't intersect the right boundary and vice versa

`coef_pyrm_down`, M^{-T} is the matrix of `basis_pyrm_up` and M^T is the matrix of `basis_pyrm_down`.

The wavelet basis is an alternate basis for V_L , but unlike the B-spline basis, it is an $L - 3$ level hierarchical basis. At level 3 there are eleven broad B-splines, and eight broad wavelets. These basis functions give the coarse description of the function. At each subsequent level going from level 3 to $L - 1$, the basis includes twice as many wavelets, and these wavelets are twice as narrow as the ones on the previous level. Each level successively adds more degrees of detail to the function.

Since each wavelet coefficients represents the amount of local detail of a particular scale, *if the function is sufficiently smooth in some region, then very few non-zero wavelet coefficients will be required in that region*³.

4.5 Scaling

One final issue is the scaling ratio between the basis functions. Traditionally [4] the wavelet functions are defined with the following scaling:

$$\begin{aligned}\phi_{i,j}(t) &= 2^{(i-L)/2} \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= 2^{(i-L)/2} \psi(2^{(i-L)}t - j)\end{aligned}\quad (18)$$

This means that at each level up, the basis functions become twice as wide, and are scaled $\frac{1}{\sqrt{2}}$ times as tall. While in many contexts this normalizing may be desirable, for optimization purposes it is counter productive. For the optimization procedure to be well conditioned [6] it is advantageous to emphasize the coarser levels and hence use the scaling defined by

$$\begin{aligned}\phi_{i,j}(t) &= 2^{L-i} \phi(2^{(i-L)}t - j) \\ \psi_{i,j}(t) &= 2^{L-i} \psi(2^{(i-L)}t - j)\end{aligned}\quad (19)$$

where the wider functions are also taller. In the pyramid code, this is achieved by multiplying all of the h and g entries by 2.

5 Implementation

The input to the wavelet spacetime problem includes the creature description, the objective function (i.e., symbolic expressions of joint torques generated from the creature description), and user defined constraints specifying desired actions (throw, catch, etc.), and inequality constraints such as joint limits on the elbow. As discussed in section 2.1, these symbolic expressions are differentiated and *compiled* into DAGs.

At this point, a constrained variational problem is defined

$$\begin{aligned}\text{minimize} & f(\Theta, \dot{\Theta}, \ddot{\Theta}) \\ \text{subject to} & C_i(\Theta, \dot{\Theta}, \ddot{\Theta}) = 0, i \in n_{eq} \\ & C_i(\Theta, \dot{\Theta}, \ddot{\Theta}) \geq 0, i \in n_{ineq}\end{aligned}\quad (20)$$

where Θ is the vector of trajectories of the degrees of freedom of the creature.

Each trajectory, $\theta(t)$, is represented in the uniform cubic B-spline basis. The unknowns are then the B-spline coefficients, \mathbf{b} , or the equivalent wavelet coefficients, \mathbf{c} , scaling the individual basis functions. This finite set of coefficients provide the information to evaluate the $\theta(t)$, $\dot{\theta}(t)$, and $\ddot{\theta}(t)$ at any time t , that comprise the leaves of the DAGs. This finite representation transforms the variational problem into a constrained non-linear optimization problem.

Non-linear optimization methods, in general, require a step that transforms the constrained problem into an unconstrained one.

Possibilities include constructing a system of Lagrange multipliers as was used in [5] or using penalty functions for the constraints [13]. In the case of penalty functions a unified *cost function* to minimize is derived as

$$F(\Theta) = f(\Theta) + \sum_{i \in n_{eq}} w_i * C_i(\Theta)^2 + \sum_{i \in n_{ineq}} w_i * (\min(C_i(\Theta), 0))^2 \quad (21)$$

where the w_i weight the individual constraints.

Solution methods then may consist of a sequence of quadratic subproblems (SQP) making a series of Newton steps towards a solution (as described in [5]). Alternatively, quasi-Newton methods such the Broyden-Fletcher-Goldfarb-Shanno (BFGS) can be used to solve the unconstrained problem [7, 13]. Quasi-Newton methods are similar to Newton's method except that the inverse of the Hessian at each iteration is approximated by a symmetric positive definite matrix, that is corrected or updated from iteration to iteration.

BFGS iterations begin with a user provided initial guess of wavelet coefficients \mathbf{c}_0 (that can be derived from B-spline coefficients using `coef_pyrm_up`) and a guess \mathbf{H}_0 of the inverse of the Hessian (usually an identity matrix leading to the first iteration being a simple gradient descent).

At each iteration, if the current solution is \mathbf{c}_k and the current guess of the inverse of the Hessian is \mathbf{H}_k , then the descent direction is $\Delta \mathbf{c}_k = -\mathbf{H}_k * \mathbf{c}_k$ and a line search finds the $\lambda_k > 0$ minimizing $F(\mathbf{c}_k + \lambda_k * \Delta \mathbf{c}_k)$. A (hopefully) better solution \mathbf{c}_{k+1} is found as $\mathbf{c}_{k+1} = \mathbf{c}_k + \lambda_k * \Delta \mathbf{c}_k$. If using the quasi-Newton method \mathbf{H}_{k+1} is updated correspondingly by using BFGS correction formula [7]

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \left(1 + \frac{\gamma^T \mathbf{H}_k \gamma}{\delta^T \gamma}\right) \frac{\delta \delta^T}{\delta^T \gamma} - \left(\frac{\delta \gamma^T \mathbf{H}_k + \mathbf{H}_k \gamma \delta^T}{\delta^T \gamma}\right) \quad (22)$$

where $\gamma = \nabla F_{k+1} - \nabla F_k$ (the change in the gradient of the cost function) and $\delta = \mathbf{c}_{k+1} - \mathbf{c}_k$ (the change in the coefficients, both B-spline and wavelet).

The newly obtained solution \mathbf{c}_{k+1} is then transformed into B-spline coefficients \mathbf{b}_{k+1} with `coef_pyrm_down`, and \mathbf{b}_{k+1} is sent to the graphical user interface for display.

If the initial function space is restricted to a coarse representation consisting of the broad B-splines and a single level of wavelets, after each iteration a simple *oracle* function adds wavelets at finer levels only when the wavelet coefficient above exceeds some tolerance. This procedure quickly approximates the optimal trajectory and smoothly converges to a final answer with sufficient detail in those regions that require it.

An important feature of the system discussed in [5] is also available in the current implementation. The user can directly modify the current solution with a simple key frame system to help *guide* the numerical process. This is critical to allow the user, for example, to move the solution from an underhand to an overhand throw, both of which represent local minima in the same optimization problem. The next iteration then begins with these new trajectories as the current guess.

6 Results

A set of experiments was run on the problem of a three-link arm and a ball (see Figure 1). The goal of the arm is to begin and end in a rest position hanging straight down, and to throw the ball into a basket. The objective function is to minimize energy, where energy is defined as the integral of the sum of the squares of the joint torques. Gravity is active.

The four graphs in Figure 6 show the convergence of five different test runs of the arm and ball example. Each plot differs only

³In this case, non-zero can be defined to be having an absolute value greater than some epsilon without incurring significant error in the representation.

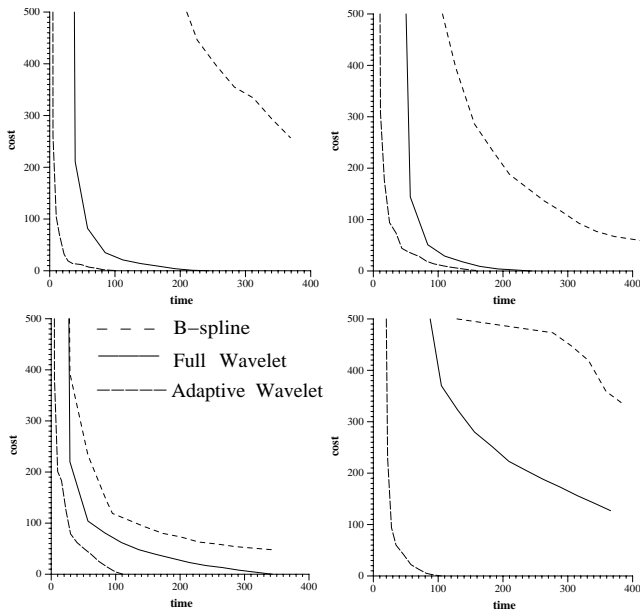


Figure 6: Convergence of Arm and Ball example for 4 different starting trajectories. The first and fourth examples resulted in underhand throws, and the rest overhand. Time is in seconds, and the cost is a weighted sum of constraint violations and energy above the local minimum.

in the starting trajectories of the arm DOF. Each run converged to either an underhand or overhand throw into the basket. The full B-spline basis contained 67 basis functions for each of the three DOF, thus there were 201 unknown coefficients to solve for. Iterations took approximately 7 seconds each on an SGI workstation with an R4000 processor. Convergence was achieved on each, but only after many iterations due to the ill-conditioning of the B-spline formulation.

The full wavelet basis also contained 67 basis function per DOF (11 B-splines at the top level and 56 wavelets below), thus iterations also took approximately the same 7 seconds. Figure 6 clearly shows the improved convergence rates of the wavelet formulations over the B-spline basis, due to better conditioned linear systems. The adaptive wavelet method with the oracle was the fastest since the number of unknowns was small in early iterations, leading to a very fast approximation of the final trajectories, in addition to the better conditioning provided by the hierarchical basis. The final few iterations involved more wavelets inserted by the oracle to complete the process. Note that in each case, a good approximation to the complete animation was achieved in less than a minute of computation.

A short sequence involving two basketball players with six degrees of freedom each (see Figures 1,7) was animated. The task was a “give and go” play. Player A passes the ball to player B, then moves towards the basket. Player B passes it back to A who makes the shot. This animation was created in stages: first player A throws the ball to a location set by the user, then player B’s actions are animated to catch the ball, then player B’s throw is animated followed by player A catching this throw and making the basket. Each stage of the animation took between 10 and 25 iterations of approximately 6-10 seconds each. The longer iteration times are due to the 6 DOF of each creature leading to twice the number of unknowns.

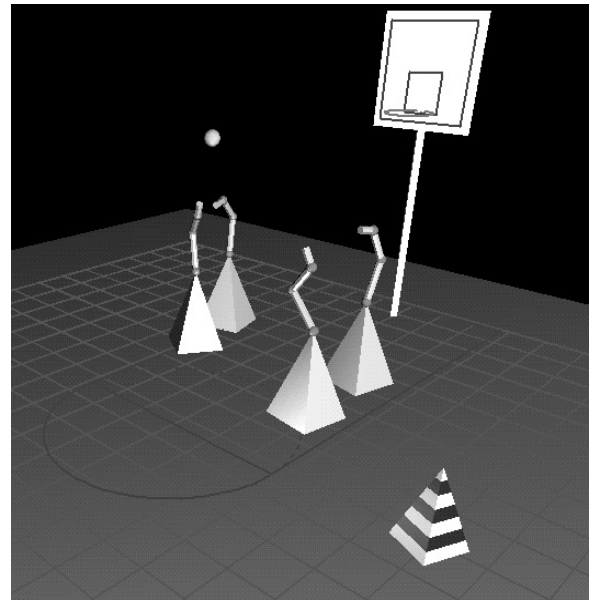


Figure 7: Scene from a basketball game.

7 Conclusion

The spacetime constraint system first suggested by Witkin and Kass [18] for animating linked figures has been shown to be an effective means of generating goal based motion. Cohen enhanced this work by demonstrating how to focus the optimization step on *windows* of spacetime and methodologies to keep the user in the optimization loop. This paper has extended this paradigm by removing two major difficulties.

The first improvement is in the runtime symbolic differentiation and subsequent *compilation* of constraints and objectives. By utilizing common subexpression elimination, a technique adopted from compiler optimization and automatic differentiation, the size and evaluation of the resulting expression trees is reduced by two orders of magnitude.

Perhaps the more important improvement lies in the representation of the trajectories of the DOF in a wavelet basis. This resulted in faster optimization iterations due to less unknown coefficients needed in smooth regions of the trajectory. In addition, even with the same number of coefficients, the systems become better conditioned and thus less iterations are required to settle to a local minimum. Results are shown for a planar three-link arm and two six DOF “basketball players”.

The paper has not discussed details of the user interface to the new spacetime system. Important aspects of the system are the ability to construct creature descriptions, specify and modify constraints and objectives and modify trajectories between optimization iterations. This requires integration of other technologies such as inverse kinematic specification, and could take advantage of high level language interfaces. These aspects of the total system are currently being investigated. The underlying mathematical framework described in this paper should now provide an excellent platform for these endeavors.

Appendix

This appendix provides pseudo code for the procedures discussed in Section 4.3. The sequence h and the sequence g are the convolution sequences used to construct the inner basis functions [4], while the vectors \underline{h}_j and the vectors \underline{g}_j are used to construct the

boundary basis functions [3, 14]. Only the vectors for the left boundary are given, the vectors for the right boundaries are the mirror images of these vectors. It is assumed that the boundary B-spline basis functions are those that arise by placing quadruple knots at the boundaries.

$$h[0..4] = \frac{1}{8} * \{1, 4, 6, 4, 1\}$$

$$g[0..10] = \frac{1}{8!} * \{-1, 124, -1677, 7904, -18482, 24264, -18482, 7904, -1677, 124, -1\}$$

$$\underline{h}_{-3} = \frac{1}{16} * \{16, 8, 0, 0, 0, \dots\}$$

$$\underline{h}_{-2} = \frac{1}{16} * \{0, 8, 12, 3, 0, 0, \dots\}$$

$$\underline{h}_{-1} = \frac{1}{16} * \{0, 0, 4, 11, 8, 2, 0, \dots\}$$

$$\frac{1}{8!} * \begin{pmatrix} \underline{g}_{-3} & \underline{g}_{-2} & \underline{g}_{-1} \\ \hline 1136914560 & -2387315040 & 123066720 \\ 27877 & 195139 & 1365937 \\ -1655323200 & 2141121840 & -2226000 \\ 27877 & 195139 & 1365937 \\ 1321223960 & 878161880 & 188417600 \\ 27877 & 195139 & 1365937 \\ -633094403 & -498772701 & -2293862247 \\ 27877 & 1365937 & -1365937 \\ 229000092 & 4726413628 & 10796596516 \\ 27877 & 195139 & 1365937 \\ -46819570 & -3606490941 & -25245248833 \\ 27877 & 195139 & 1365937 \\ \hline 124 & 7904 & 24264 \\ -1 & -1677 & -18482 \\ 0 & 124 & 7904 \\ . & -1 & -1677 \\ . & 0 & 124 \\ . & . & -1 \\ . & . & 0 \\ . & . & . \end{pmatrix}$$

The following procedure describes how the two-part basis functions are constructed by linearly combining the B-spline basis functions. (see Equations 5 and 6).

```
basis_xform_up( b_in [], b_out [], w_out [], L )
  b_out = w_out = 0 ; /* zero vectors */
  for ( j = 0 ; j ≤ 2L-1 - 4 ; j++ )
    for ( k = 2j ; k ≤ (2j + 4) ; k++ )
      b_out[j] += h[k - 2j] * b_in[k] ;
  for ( j = 0 ; j ≤ 2L-1 - 7 ; j++ )
    for ( k = 2j ; k ≤ (2j + 10) ; k++ )
      w_out[j] += g[k - 2j] * b_in[k] ;
  for ( j in [-3, -2, -1, 2L-1 - 3, 2L-1 - 2, 2L-1 - 1] )
    b_out[j] = hj · b_in ; /*dot product*/
  for ( j in [-3, -2, -1, 2L-1 - 4, 2L-1 - 5, 2L-1 - 6] )
    w_out[j] = gj · b_in ;
```

This procedure can be expressed as multiplication by a banded matrix, and so the inverse procedure **basis_xform_down**(*b_in* [], *w_in* [], *b_out* [], *L*) can be obtained by solving this banded linear system.

The following procedure can be used to obtain B-spline coefficients given two-part coefficients.

```
coef_xform_down( b_in [], w_in [], b_out [], L )
  b_out = 0 ; /* zero vector */
  for ( k = 0 ; k ≤ 2L-1 - 4 ; k++ )
    for ( j = 2k ; j ≤ (2k + 4) ; j++ )
      b_out[j] += h[j - 2k] * b_in[k] ;
  for ( k = 0 ; k ≤ 2L-1 - 7 ; k++ )
```

```
    for ( j = 2k ; j ≤ (2k + 10) ; j++ )
      b_out[j] += g[j - 2k] * w_in[k] ;
  for ( k in [-3, -2, -1, 2L-1 - 3, 2L-1 - 2, 2L-1 - 1] )
    b_out += hk * b_in[k] ; /*vector addition*/
  for ( k in [-3, -2, -1, 2L-1 - 4, 2L-1 - 5, 2L-1 - 6] )
    b_out += gk * w_in[k] ;
```

This procedure can be expressed as multiplication by a banded matrix, and so the inverse procedure **coef_xform_up**(*b_in* [], *b_out* [], *w_out* [], *L*) can be obtained by solving this banded linear system.

Acknowledgements

The authors owe a debt to Charles Rose who implemented the user interface for this work. This research was supported in part by the National Science Foundation, Grant CCR-9296146.

References

- [1] AHO, A. V., SETHI, R., AND ULLMAN, J. D. *Compilers: Principles, Techniques and Tools*. Addison Wesley, 1986.
- [2] BARTELS, R., BEATTY, J., AND BARSKY, B. *An Introduction to Splines for Use in Computer Graphics and Modeling*. Morgan Kaufmann, 1987.
- [3] CHUI, C., AND QUAK, E. Wavelets on a bounded interval. *Numerical Methods of Approximation Theory 9* (1992), 53–75.
- [4] CHUI, C. K. *An Introduction to Wavelets*, vol. 1 of *Wavelet Analysis and its Applications*. Academic Press Inc., 1992.
- [5] COHEN, M. F. Interactive spacetime control for animation. *Computer Graphics 26*, 2 (July 1992), 293–302.
- [6] DAHMEN, W., AND KUNOTH, A. Multilevel preconditioning. *Numerische Mathematik 63* (1992), 315–344.
- [7] FLETCHER, R. *Practical Methods of Optimization*, vol. 1. John Wiley and Sons, 1980.
- [8] FORSEY, D., AND BARTELS, R. Hierarchical b-spline refinement. *Computer Graphics 22*, 4 (August 1988), 205–212.
- [9] GORTLER, S., AND COHEN, M. F. Variational modeling with wavelets. Tech. Rep. CS-TR-456-94, Department of Computer Science, Princeton University, 1994.
- [10] GORTLER, S., SCHRÖDER, P., COHEN, M., AND HANRAHAN, P. Wavelet radiosity. In *Computer Graphics, Annual Conference Series, 1993* (August 1993), Siggraph, pp. 221–230.
- [11] KASS, M. Condor: Constraint based dataflow. *Computer Graphics 26*, 2 (July 1992), 321–330.
- [12] NGO, J. T., AND MARKS, J. Spacetime constraints revisited. In *Computer Graphics, Annual Conference Series, 1993* (August 1993), Siggraph, pp. 343–350.
- [13] PAPALAMBROS, P. Y., AND WILDE, D. J. *Principles of Optimal Design*. Cambridge University Press, Cambridge, England, 1988.
- [14] QUAK, E., AND WEYRICH, N. Decomposition and reconstruction algorithms for spline wavelets on a bounded interval. Tech. Rep. 294, Center for Approximation Theory, Texas A&M, 1993.
- [15] RALL, L. B. *Automatic Differentiation: Techniques and Applications*. Springer-Verlag, 1981.
- [16] TERZOPOULOS, D. Image analysis using multigrid relaxation methods. *IEEE PAMI 8*, 2 (March 1986), 129–139.
- [17] VAN DE PANNE, M., AND FIUME, E. Sensor-actuator networks. In *Computer Graphics, Annual Conference Series, 1993* (August 1993), Siggraph, pp. 335–342.
- [18] WITKIN, A., AND KASS, M. Spacetime constraints. *Computer Graphics 22*, 4 (August 1988), 159–168.