Variational Geometric Modeling with Wavelets

Steven J. Gortler and Michael F. Cohen
Microsoft Research
Redmond, WA

(excerpted from "Hierarchical and Variational Geometric Modeling with Wavelets", by Steven J. Gortler and Michael F. Cohen, Princeton University, presented at the 1995 Symposium on Interactive 3D Graphics)

Abstract

This portion of the notes discusses how wavelet techniques may be applied to a variety of geometric modeling tools. In particular, wavelet decompositions are shown to be useful for hierarchical control point or least squares editing. In addition, direct curve and surface manipulation methods using an underlying geometric variational principle can be solved more efficiently by using a wavelet basis. Because the wavelet basis is hierarchical, iterative solution methods converge rapidly. Also, since the wavelet coefficients indicate the degree of detail in the solution, the number of basis functions needed to express the variational minimum can be reduced, avoiding unnecessary computation. An implementation of a curve and surface modeler based on these ideas is discussed and experimental results are reported.

1 Introduction

Wavelet analysis provides a set of tools for representing functions hierarchically. These tools can be used to facilitate a number of geometric modeling operations easily and efficiently. In particular, these notes outline three paradigms for free-form curve and surface construction: control point editing, direct manipulation using least squares, and direct manipulation using variational minimization techniques. For each of these paradigms, the hierarchical nature of wavelet analysis can be used to either provide a more intuitive modeling interface or to provide more efficient numerical solutions.

In control point editing, the user sculpts a free-form curve or surface by dragging a set of control points. A better interface allows the user to directly manipulate the curve or surface itself, which defines a set of constraints. In a *least squares* paradigm, given a current curve or surface, the modeling tool returns the curve or surface that meets the constraints by changing the current control points by the least squares amount [1, 13].

The behavior of the modeling tool is determined by the type of control points and *basis functions* used to describe the curve or surface. With the uniform cubic B-spline basis, for example, the user's actions result in local changes at a predetermined scale. This is not fully desirable; at times the user may want to make fine changes of detail, while at other times he may want to easily make broad changes. Hierarchical B-splines offer a representation that allows both control point and least squares editing to be done at multiple resolutions [11]. Hierarchical B-splines, though, form an over-representation for curves and surface (i.e., any curve has multiple representations using hierarchical B-splines). As a result, the same curve may behave differently to a user depending on the particular underlying representation. In contrast, B-spline wavelets form a hierarchical basis for the space of B-spline curves and surfaces in which every object has a unique representation. Wavelet methods in conjunction with hierarchical B-splines provide a method for constructing a useful geometric modeling interface. This approach is similar to the one described by Finkelstein and Salesin [9]. In these notes we will discuss some of the various issues that are relevant to building such a modeling tool.

Variational modeling is a third general paradigm for geometric modeling[3, 35, 24]. In this setting, a user alters a curve or surface by directly manipulation, as above, defining a set of constraints. The variational modeling paradigm seeks the "best" solution amongst all answers that meet the constraints. The notion of best, which is formally defined as the solution that *minimizes some energy function*, is often taken to mean the *smoothest* solution.

In theory, the desired solution is the curve or surface that has the minimum energy of *all* possible curves or surfaces that meet the constraints. Unfortunately there is little hope to find a closed form solution [1]. Therefore, in practice, the

[1] But see [23].

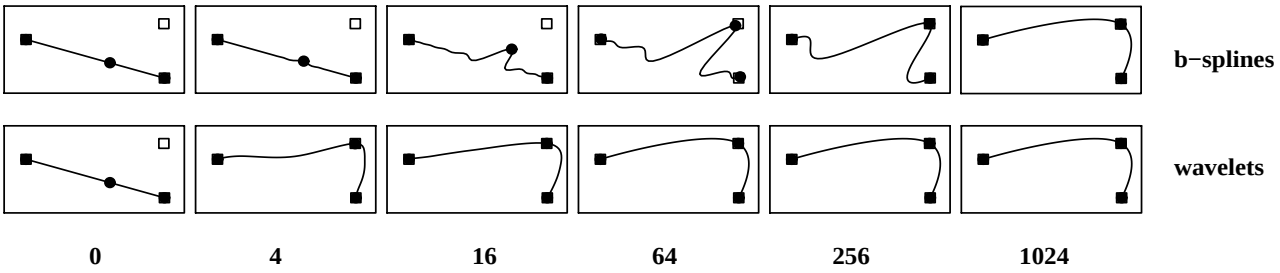

Figure 1: Minimum energy solutions subject to three constraints, found by the B-spline and wavelet methods after various numbers (0-1024) of iterations. (65 variables, 3 constraints). This illustrates the ill conditioning of the B-spline optimization problem.

"space" of parametric curves or surfaces is restricted to those represented by a linear combination of a fixed set of basis functions such as cubic B-splines. Given a set of n basis functions, the goal of finding the best curve or surface is then reduced to that of finding the best set of n coefficients. This reduction is referred to as the *finite element method* [33].

The general case requires solving a non-linear optimization problem. In the best case, the energy function is quadratic and the constraints are linear leading to a single linear system to solve. But even this can be costly when n is large since direct methods for matrix inversion require $O(n^3)$ time. To accelerate this process it is tempting to use gradient-type iterative methods to solve the linear system; these methods only take $O(n)$ time per iteration, due to the $O(n)$ matrix sparsity created by the finite element formulation. Unfortunately, the linear systems arising from a finite element formulation are often expensive to solve using iterative methods. This is because the systems are ill-conditioned, and thus require many iterations to converge to a minimum [31, 30]. Intuitively speaking this occurs because each basis function represents a very narrow region of the answer; there is no basis function which can be moved to change the answer in some broad manner. For example, changing one coefficient in a cubic B-spline curve during an iteration alters the curvature in a local region only. In order to produce a broad smooth curve, the coefficients of the neighboring B-splines will move in next few iterations. Over the next many iterations, the solution process will affect wider and wider regions, and the effect will spread out slowly like a wave moving along a string. The result is very slow convergence (see Figure (1)). One method used to combat this problem is multigridding [31, 12], where a sequence of problems at different resolution levels are posed and solved.

An alternative approach, is to use a *wavelet* basis instead of a standard finite element basis [30, 27, 18, 26]. In a wavelet basis, the answer is represented hierarchically. This allows the solution method to alter the answer at any desired resolution by altering the proper basis function, and thus the ill-conditioning is avoided. We will show how to use a wavelet construction, which is based on cubic B-splines, to quickly solve variational modeling problems in an elegant fashion.

Another problem with the finite element approach is choosing the density of the basis functions. If too few basis functions (too few B-spline segments or tensor product B-spline patches) are used then the solution obtained will be far from the actual minimum. If too many basis functions are used then unnecessary computation will be performed during each iteration (n is too big). In order to successfully choose a proper density, one must know how much detail exists in the variational minimum answer. Since, a priori, this is unknown, an efficient solver must be able to adaptively change the basis during the solution process [35], one needs an easy way to detect that too many or too few basis functions are being used. In addition, one needs a basis for which adding more detail, (i.e., refinement), is easy. Wavelets offer a basis where this task can be accomplished quickly and elegantly.

The work presented here combines the wavelet approaches of [30], [15], and [19]. Like [30], we use hierarchical basis functions as a pre-conditioner, so that fewer iterations are needed for convergence. Similar to [15] and [19], wavelets are also used as a method for limiting the solution method to the proper level of detail.

2 Geometric Modeling with Wavelets

The styles of interactive control discussed in the introduction will be revisited in the context of hierarchical representations. *Multiresolution modeling* allows the user to interactively modify the curve or surface at different resolution levels. This allows the user to make broad changes while maintaining the details, and conversely detailed changes while maintaining the overall shape. Two types of hierarchical manipulation are considered, control point dragging and a direct manipulation involving solving a least squares problem.

In contrast, *variational modeling* allows the user to directly manipulate the curve or surface with the curve or surface maintaining some notion of overall smoothness subject to user imposed constraints. This physically based paradigm provides an intuitive means for shape control. Each of these paradigms will be explored in the context of wavelet bases which will be shown to provide the required hooks for such interaction and/or significant computational savings.

2.1 Multiresolution Modeling

A multiresolution representation such as a hierarchical B-spline or wavelet representation may be used to implement a multiresolution modeling system. This section explores the choices that must be made when designing a multiresolution tool. Two related methods are described; direct control point manipulation and a least squares solver.

In control point modeling, the user is allowed to directly alter the coefficient values, by clicking and dragging on control points. In the least squares scheme [1, 13], the user can click and drag directly on the curve or surface, defining interpolation and tangent constraints, linear with respect to the control points. The system returns the curve or surface that satisfies these linear constraints, by changing the coefficients by the least squares amount. Least square solutions can be found very inexpensively using the pseudoinverse [13]. The least squared problem can also be posed as a minimization problem [35], whose solution can be found by solving a sparse, well conditioned, linear system.

In multiresolution versions of these two schemes, the user chooses the resolution level i, and then only the quantities of basis functions on level i are altered. The locality of the effect on the curve or surface is directly tied to the chosen level i. In control point modeling, the control polygon at level i is manipulated by the user. In a least squares scheme, the user is provided a direct handle on the curve or surface itself, and the least squares solution is found only using the basis functions on level i. The least-squares approach offers a much more intuitive interface, and (for curves) works at interactive speeds.

One decision to be made is whether to expose the user to hierarchical B-splines or to wavelets. It is easy to see that manipulating wavelet basis functions does not produce an intuitive interface. Moving such a control point, and thus changing the amount of some wavelet basis function used, changes the solution in a "wave" like fashion. In contrast, it is more intuitive to move a B-spline control point which changes the solution in a "hump" like fashion. Thus the user in this case should manipulate the hierarchical B-spline functions.

2.1.1 Projections between Levels

An important tool to implement the ideas in these notes is the ability to find the closet (in some sense) lower resolution curve or surface to one constructed at a higher resolution. This process is called *projection*. The inverse process, *refinement*, takes a low resolution curve or surface and adds additional degrees of freedom, in general without changing the shape.

There are many ways to obtain a lower resolution version of some object. For example, given an object at some resolution of detail, one could obtain a lower resolution version by throwing away every other control point. Subsampling is not a true projection; starting with a smooth curve and then expressing that smooth curve in the higher resolution B-spline basis basis and finally subsampling the control points will not return the original smooth curve we began with.

Another way of obtaining a smoothed version of the object is by *orthogonally* projecting the object from a space defined by a larger set of basis functions to a smaller (i.e., lower resolution) space spanning linear combinations of fewer basis functions. The orthogonal projection is the object in the lower resolution space that is closest to object in the higher resolution space using the L^2 measure. In general, this involves a sort of low-pass filtering. This is the

approach used in [9]. Although this is a very elegant way of obtaining a lower resolution version of an object, it has a few drawbacks. The particular filter sequence used is infinite in length (although it does decay rapidly from its centers) and so performing this task efficiently can be troublesome. Also, because these sequences are not local, then a single change to one B-spline coefficient at some level will alter all of the coefficients of the projection at the next courser level.

One good compromise between these two extremes (subsampling, and orthogonal projection), is to use the filter sequence given for the non-orthogonal wavelet construction by Cohen et al. [6]. This projection in non-orthogonal, but it is entirely local. This is the choice we have used in our multiresolution modeling tool.

2.1.2 Representing Detail

When one projects a curve or surface to a lower resolution, the detail may be lost. One can, however, explicitly store this lost detail, perhaps to be added back in later, or to be added to another curve or surface to give it a similar quality.

What set of basis functions should be used to represent the detail. If a wavelet projection is used to define the lower resolution versions of the object, then the detail can be represented by using the corresponding wavelet functions. The other option is to represent the detail using hierarchical B-spline functions. The disadvantage of using hierarchical B-splines is that there are roughly $2n$ B-splines in the hierarchy, and only n wavelets.

The advantage of using hierarchical B-splines however is that they maintain the relationship between the detail and the local orientation (captured by the local tangent, normal, and binormal frame) better. When the user changes the broad sweep of the curve, changing the orientation, the detail functions are remixed. If the detail functions are wavelet functions, then changing the normal and tangent frame remixes "wave" shaped functions introducing non-intuitive wiggles. If the detail functions are B-spline basis functions, then "hump" shaped functions get remixed, yieding more intuitive changes. Also if the detail functions are B-splines, then because there are twice as many B-splines than wavelets, the tangent and normal directions are computed at twice as many sample points allowing the detail to follow the orientation with more fidelity.

3 Variational Modeling

The variational modeling paradigm generalizes the least squares notion to any *objective* function minimization, typically one representing minimizing curvature. The variational problem leads to a non-linear optimization problem over a finite set of variables when cast into a given basis.

There are a variety of objective functions used in geometric modeling [24, 29] In our implementation we have used the *thin-plate* measure which is based on minimizing the parametric second derivatives [33, 3, 35]. If the vector of unknown coefficients are denoted by $\mathbf{x}$, and the linear position and tangent constraints imposed by the user's action are given by the set of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, then the thin plate minimum may be found by solving the following linear system [35].

$$\begin{vmatrix} \mathbf{H} & \mathbf{A}^{\mathbf{T}} \\ \mathbf{A} & \mathbf{0} \end{vmatrix} \begin{vmatrix} \mathbf{x} \\ \lambda \end{vmatrix} = \begin{vmatrix} \mathbf{0} \\ \mathbf{b} \end{vmatrix} \tag{1}$$

Where $\mathbf{H}$ is the Hessian matrix defined by the thin plate functional, and λ are Lagrange variables.

3.1 Hierarchical Conditioning

Wavelets can be used in the context of variational modeling so that the solution may be obtained more efficiently.

In the B-spline basis, the optimization procedure resulted in the linear system given by Equation (1). In the wavelet basis, a different linear system results. If the wavelet filter sequences defining the wavelet transform are contained in the matrix $\mathbf{W}$, then an equivalent linear system is given by

$$\begin{vmatrix} \bar{\mathbf{H}} & \bar{\mathbf{A}}^{T} \\ \bar{\mathbf{A}} & \mathbf{0} \end{vmatrix} \begin{vmatrix} \bar{\mathbf{x}} \\ \lambda \end{vmatrix} = \begin{vmatrix} \mathbf{0} \\ \mathbf{b} \end{vmatrix} \tag{2}$$

where the bars signify that the variables are wavelet coefficients, $\bar{\mathbf{x}} = \mathbf{W}\mathbf{x}$, and the Hessian and constraint matrix are expressed with respect to the wavelet basis. To see the relationship with the B-spline system, the new system can also be written down as

$$\begin{vmatrix} \mathbf{W^{-T}H W^{-1}} & \mathbf{W^{-T}A^T} \\ \mathbf{AW^{-1}} & \mathbf{0} \end{vmatrix} \begin{vmatrix} \bar{\mathbf{x}} \\ \lambda \end{vmatrix} = \begin{vmatrix} \mathbf{0} \\ \mathbf{b} \end{vmatrix} \tag{3}$$

Although Equation (1) and Equation (2/3) imply each other, they are two distinct linear systems of equations. Because the wavelet system (2/3) is hierarchical it will not suffer from the poor conditioning of the B-spline system of Equation (1). For a rigorous discussion of the relevant theory see [8].

The scaling of the basis functions is very significant for the behavior of the optimizing procedures. Traditionally the wavelet functions are defined with the scaling defined in [22, 26]. At each level moving 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 [18, 8] it is essential to emphasize the coarser levels. The correct theoretical scaling depends on both the energy function used, and the dimension of problem. For a fuller discussion, see the Appendix in [16]. In the experiments described below a different scaling was used.

As one goes one level down, the basis functions become twice as wide, and $1/2$ as tall. In the pyramid code, this is achieved by multiplying all of the scaling and wavelet filter coefficients by 2, and all of the dual coefficients by $1/2$ The proper scaling is essential to obtain the quick convergence of the wavelet method when steepest descent or conjugate gradient iteration is used. Scaling is not important with Gauss-Seidel iteration, which will perform the same sequence of iterations regardless of scale.

3.1.1 Explicit vs. Implicit

There is now a choice to make. In an iterative conjugate gradient solver, the common operation is multiplication of a vector times the wavelet matrix given in Equations (2/3). There are two ways to implement this.

One approach, the *explicit* approach, is to compute and store the wavelet Hessian matrix $\bar{\mathbf{H}}$ and the wavelet constraint matrix $\bar{\mathbf{A}}$ (Equation (2)). These can be computed directly from a closed form (piecewise polynomial) representation of the wavelet functions. Unfortunately, these matrices are not as sparse as the B-spline Hessian and constraint matrices.

Alternatively, there is the *implicit* approach [37, 30] which only computes and stores the entries of the B-spline matrices $\mathbf{H}$ and $\mathbf{A}$ (Equation (3)). Multiplication by the $\mathbf{W}$ matrices is accomplished using a linear time pyramid transform procedure.

The advantage of this approach is that the whole multiply remains $O(n)$ in both time and space, since the pyramid procedures run in linear time, and the matrices $\mathbf{H}$ and $\mathbf{A}$ are $O(n)$ sparse. Even though one of the methods explicitly uses wavelet terms while the other uses B-spline terms, these two methods are mathematically equivalent, and so both will have the same condition properties.

3.2 Adaptive Oracle

By limiting the possible surfaces to only those that can be expressed as a linear combination of a fixed set of basis functions, one obtains an approximation of the true optimal surface. As more basis functions are added, the space of possible solutions becomes richer and a closer approximation to the true optimal surface can be made. Unfortunately, as the space becomes richer, the number of unknown coefficients increases, and thus the amount of computation required per iteration grows. A priori, it is unknown how many basis functions are needed. Thus, it is desirable to have a solution method that adaptively chooses the appropriate basis functions. This approach was applied using hierarchical B-splines in [35]. When refinement was necessary, "thinner" B-splines basis functions were added, and the redundant original "wider" B-splines were removed. With wavelets, all that must be done is to add in new "thinner" wavelets wherever refinement is deemed necessary. Since the wavelets coefficients correspond directly to local detail, all previously computed coefficients are still valid.

The decision process of what particular wavelets to add and remove is governed by an `oracle` procedure which is called after every fixed number of iterations. The oracle must decide what level of detail is required in each region of the curve or surface.

When some region of the solution does not need fine detail, the corresponding wavelet coefficients are near zero, and so the first thing the `oracle` does is to deactivate the wavelet basis functions whose corresponding coefficients are below some small threshold. The `oracle` then activates new wavelet basis functions where it feels more detail may be needed. There are two criteria used. If a constraint is not being met, then the oracle adds in finer wavelet functions in the region that is closest in parameter space to the unmet constraint. Even if all the constraints are being met, it is possible that more basis functions would allow the freedom to find a solution with lower energy. This is accomplished by activating finer basis functions near those with coefficients above some maximum threshold.

To avoid cycles, a basis function is marked as being `dormant` when it is removed from consideration. Of course, it is possible that later on the solution may really need this basis function, and so periodically there is a `revival` phase, where the `dormant` marks are removed.

3.3 User Interface

A user of the system is first presented with a default curve or surface. Constraints can then be introduced by clicking on the curve or surface with the mouse. The location of the mouse click defines a parametric position t (and s) on the curve (or surface). The user can then drag this point to a new location to define an interpolation constraint. Tangent constraints at a point can also be defined by orienting "arrow" icons at the point. Once the constraint is set, the solver is called to compute the minimum energy solution that satisfies the constraints placed so far and the result is displayed.

When the solution is completed, the result provides information for not only the curve or surface satisfying the specific value of the new constraint, but for all curves or surfaces with respect to any value of this constraint. Once the linear system (Equation (2/3)) with the newest constraint has been solved, the solver stores the delta vector

$$\frac{\mathbf{\Delta \bar{x}}}{\mathbf{\Delta b_m}} \tag{4}$$

where m is the index of the newest constraint, and b_m is the constraint value (i.e., the position or tangent specified by the user). This vector stores the change of the coefficient vector due to a unit change in the new constraint $\mathbf{\Delta b_m}$, essentially a column of the inverse matrix. The user is now free to interactively move the target location of the constraint without having to resolve the system since, as long as the parameters s, and t of the constraints do not change, the matrix of the system, and thus its inverse, do not change. However, as soon as a new constraint is added (or a change to the parameters s and t is made) there is fresh linear system that must be solved, and all of the delta vectors are invalidated. The ability to interactively change the value of a constraint is indicated to the user by coloring the constraint icon.

3.4 Variational Modeling Results

A series of experiments were conducted to examine the performance of the wavelet based system compared to a B-spline basis. In the curve experiments, the number of levels of the hierarchy, L, was fixed to 6, and in the surface experiments, L was fixed as 5. The optimization process was then run on problems with different numbers of constraints. The results of these tests are shown in Figures 2 and 3. These graphs show the convergence behavior of three different methods, solving with the complete B-spline basis, solving with the complete wavelet basis, and solving with an adaptive wavelet basis that uses an oracle. (The wavelet results shown here are using the *implicit* implementation). If $\mathbf{x^{(m)}}$ is the computed solution expressed as B-spline coefficients at time m, and $\mathbf{x}^*$ is the correct solution of the complete linear system [2] (i.e., the complete system with $2^L + 1$ variables, and no adaptive oracle being used) then the error at time m is defined as

$$\frac{\sum_j \mid x_j^* - x_j^{(m)} \mid}{\sum_j \mid x_j^* - x_j^{(0)} \mid} \tag{5}$$

[2] computed numerically to high accuracy

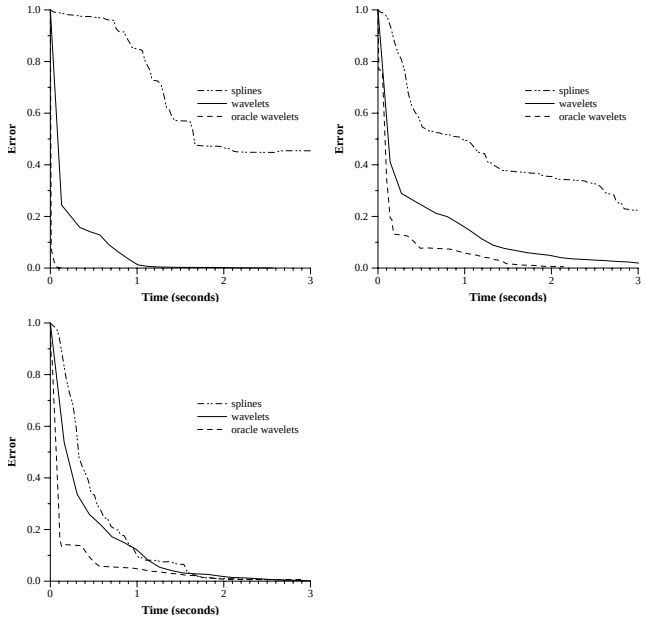

Figure 2: Error per time. Curve with 65 control points, 3, 7, and 13 constraints.

To obtain the starting condition $\mathbf{x}^{(\mathbf{0})}$, two constraints were initialized at the ends of the curve, and the minimal thin plate solution (which in this case is a straight line) was computed. (For surfaces, the four corners were constrained.) All times were taken from runs on an SGI R4000 reality engine. [3]

When there are large gaps between the constraints, the B-spline method is very poorly conditioned, and converges quite slowly while the wavelet method converges dramatically faster. In these problems, the oracle decides that it needs only a very small active set of wavelets and so the adaptive method converges even faster. As the number of constraints is increased, the solution becomes more tightly constrained, and the condition of the B-spline system improves. (Just by satisfying the constraints, the B-spline solution is very close to minimal energy). Meanwhile the oracle requires a larger active set of wavelets. Eventually, when enough constraints are present, the wavelet methods no longer offer an advantage over B-splines.

Experiments were also run where all the constraints were along the boundary of the surface. In these experiments there are many constraints, but the since the constraints are along the boundary, much of the surface is "distant" from any constraint. In these problems, the wavelets also performed much better than the B-spline method.

4 Conclusion

These notes have explored the use of wavelet analysis in a variety of modeling settings. It has shown how wavelets can be used to obtain multiresolution control point and least squares control. It has shown how wavelets can be used to solve variational problems more efficiently.

Future work will be required to explore the use of higher order functionals like those given in [24, 29]. Because the optimization problems resulting from those functionals are non-linear, they are much more computationally expensive, and it is even more important to find efficient methods. It is also important to study optimization modeling methods where constraint changes only have local effects.

Many of these concepts can be extended beyond the realm of tensor product uniform B-splines. Just as one can

[3] In the curve experiments, each B-spline iteration took 0.0035 seconds, while each iteration of the implicit wavelet method took 0.011 seconds. For the surface experiments, each B-spline iteration took 0.68 seconds while each iteration of the implicit wavelet method took 0.85 seconds. (The wavelet iterations using the explicit representation took about 10 times as long).

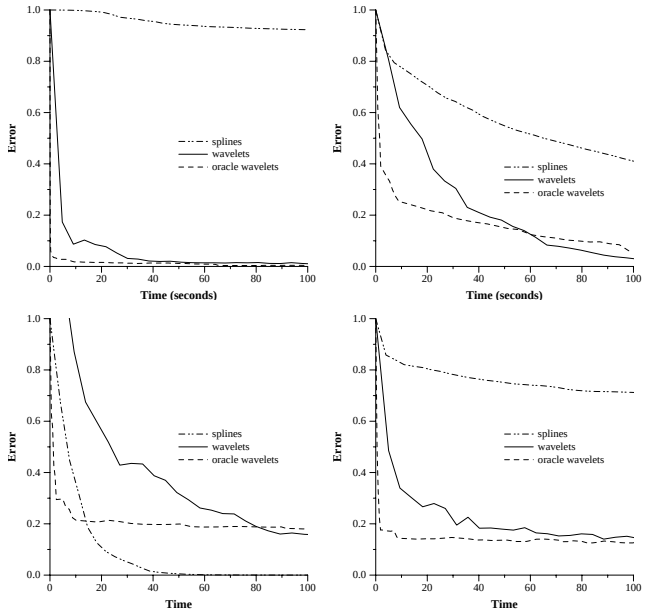

Figure 3: Error per time. Surface with 1089 control points, 11,23,64 evenly space constraints, and 62 constraints along the boundary.

create a ladder of nested function spaces using uniform cubic B-splines of various resolutions, one can also create a nested ladder using non-uniform B-splines [21].

Subdivision surfaces are a powerful technique for describing surfaces with arbitrary topology [17]. A subdivision surface is defined by iteratively refining an input control mesh. As explained by Lounsbery et al. [20], one can develop a wavelet decomposition of such surfaces. Thus, many of the ideas developed above may be applicable to that representation as well.

Acknowledgements

We are grateful to James H. Shaw for developing the graphical interface to the optimization program.