Multiresolution for Curves and Surfaces Based On Constraining Wavelets

L. Olsen [a,*], F.F. Samavati [a], R.H. Bartels [b]

[a] *Department of Computer Science, University of Calgary*
[b] *Department of Computer Science, University of Waterloo*

Abstract

We present a novel method for determining local multiresolution filters for a broad range of subdivision schemes. Our approach is based on constraining the wavelet coefficients such that the coefficients at even vertices can be computed from the coefficients of neighboring odd vertices. This constraint leads to an initial set of decomposition filters. To increase the quality of these initial filters, we use an optimization that reduces the size of the wavelet coefficients. The resulting multiresolution filters yield a biorthogonal wavelet system whose construction is similar to the lifting scheme. This approach is demonstrated in depth for cubic B-spline curves and Loop subdivision surfaces. Our filters are shown to perform comparably with existing filters.

Key words: multiresolution, splines, curve and surface representations, object heirarchies, geometric algorithms

1 Introduction

Subdivision curves and surfaces are widely used to model graphical objects. An artist or modeler only needs to construct a coarse representation of an object, which can then be enhanced by subdivision to reach a smooth representation. Subdivision schemes that work on arbitrary polygonal meshes or offer extensions to traditional subdivision such as sharp creases, are even more useful to a modeler.

* Corresponding author.
Email address: olsenl@cpsc.ucalgary.ca (L. Olsen).

Where subdivision offers an increase in the resolution of a model, multiresolution introduces an analogous decrease in resolution via a decomposition process. Trivially, a model that is the product of subdivision can be decomposed without introducing any errors. The more interesting use of multiresolution is to produce a low-resolution approximation of a high-resolution model that is not the product of subdivision, but that has the topology of a subdivided model. To allow for reconstruction of the original model, the multiresolution process must determine and store some error terms at every level of the decomposition. There are many potential applications of multiresolution, of which two popular ones are mesh editing and mesh compression.

We present a method for constructing multiresolution filters for curves and surfaces based on constraining the wavelets. Starting from a parameterized relationship between an even wavelet coefficient and a neighborhood of odd wavelet coefficients, we can simultaneously determine a way to compute even wavelets from the odd neighbors *and* a decomposition mask for replacing even vertices with a coarse approximation. The errors can then be used in an optimization step to reach a better coarse approximation.

This paper contributes a novel numerical approach for constructing local multiresolution filters for any subdivision scheme. Our method is easy to understand and apply, yet also fits in nicely with previous work on wavelets. In fact, our filters yield a biorthogonal wavelet system. To demonstrate our approach, we have developed multiresolution settings for cubic B-spline curves, as well as Loop surfaces with irregular connectivity. In both cases, we also consider boundary conditions.

The paper is organized as follows: Section 2 presents notation that will be used for developing the rest of the paper; Section 3 discusses the existing work on multiresolution; Section 4 provides a general overview of our approach, while Sections 5 and 6 demonstrate our approach in detail for 3^{rd}-degree (cubic) B-spline curves and Loop surfaces. Finally, Section 7 discusses future research directions for this work.

2 Notation

Subdivision is said to produce fine data from coarse data. The position of any fine vertex is computed as a linear combination of coarse vertices, meaning that it is feasible to express subdivision in matrix form. Furthermore, subdivision is a local operation: the position of a fine vertex is determined by a linear combination of a local neighborhood of coarse vertices. This results in subdivision matrices that are banded and that contain repetitive entries, which in turn implies that subdivision can be implemented in linear time.

The subdivision rules for a particular scheme usually come in two flavors. The *even* rule is used to displace existing vertices based on a local neighborhood, while the *odd* rule creates new vertices, usually along the edges between existing vertices. This terminology stems from the way subdivision curves are indexed, but we use it here in a more general sense, where even implies that a vertex exists at a coarser level and odd implies that it does not.

Multiresolution encapsulates two processes: decomposition (reverse subdivision plus error representation), and reconstruction (subdivision plus error correction). These processes can be formalized with matrix notation. Denote coarse data as $\mathbf{C} = [c_0\ c_1\ \ldots]^T$ and fine data as $\mathbf{F} = [f_0\ f_1\ \ldots]^T$, where c_i and f_i represent vertices in $\Re^n$.

Decomposition is the process of downsampling some fine data $\mathbf{F}$ into a coarse approximation $\mathbf{C}$ and some extra information, the *wavelet coefficients* $\mathbf{D} = [d_0\ d_1\ \ldots]^T$, where d_i are vector quantities. Wavelet coefficients are more generically referred to as *details*, a term that we will use throughout this paper. The details allow the original fine data to be reconstructed. Formally, we can say that

$$\begin{aligned}\mathbf{C} &= \mathbf{A}\mathbf{F}\\ \mathbf{D} &= \mathbf{B}\mathbf{F}\ ,\end{aligned}$$

where $\mathbf{A}$ and $\mathbf{B}$ are wide matrices known as *decomposition filters*.

Reconstruction is the process of increasing the resolution of coarse data $\mathbf{C}$ to produce fine data $\mathbf{F}$; if there are details $\mathbf{D}$ available, the reconstruction process can use them to exactly reproduce the original fine data or to introduce high-frequency information. Formally, we can write

$$\mathbf{F} = \mathbf{P}\mathbf{C} + \mathbf{Q}\mathbf{D}\ ,$$

where $\mathbf{P}$ and $\mathbf{Q}$ are tall matrices known as *reconstruction filters*.

Together, $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{A}$, and $\mathbf{B}$ are known as the *multiresolution filters*. Reconstruction and decomposition are inverse processes, thus

$$\left[\begin{array}{c}\mathbf{A}\\ \hline \mathbf{B}\end{array}\right][\mathbf{P}|\mathbf{Q}] = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} = \mathbf{I}\ . \tag{1}$$

For 1D objects without boundaries, the multiresolution filters can be entirely described by finite sequences representing regular columns or rows of the corresponding matrices. Reconstruction filters $\mathbf{P}$ and $\mathbf{Q}$ will consist of shifted

versions of a regular column, the elements of which can be represented by the sequences $(\ldots, p_{-1}, p_0, p_1, \ldots)$ and $(\ldots, q_{-1}, q_0, q_1, \ldots)$ respectively; these sequences are called *reconstruction masks*. (Note that most subdivision schemes are symmetric, in which cases $p_{-i} = p_i$ and $q_{-i} = q_i$.)

Decomposition filters also often contain repetitive shifted entries, but instead of the columns, it is the rows that are repeated. The non-zero entries $(\ldots, a_{-1}, a_0, a_1, \ldots)$ and $(\ldots, b_{-1}, b_0, b_1, \ldots)$ of typical rows in $\mathbf{A}$ and $\mathbf{B}$, respectively, are called *decomposition masks*.

For higher-dimensional objects such as polygonal meshes, there is less structure in the multiresolution filters, and thus repetitive rows and columns are generally not observed. However, the behaviors of decomposition and reconstruction can still usually be represented more compactly by looking at the local interactions of vertices. A mask can then be considered, more generally, to refer to compact (non-matrix) representations of multiresolution filters.

3 Previous Work

3.1 Subdivision

Parametric curves are a useful in computer graphics for interactive modeling. A parametric curve $Q(u) = \sum_i C_i B_i(u)$ is defined by an ordered set of control vertices $C = \{c_0, c_1, \ldots, c_n\}$ and a set of blending functions B; the blending functions determine how much each control vertex contributes to a given point along the curve, i.e. *blend* the control points into curve samples.

Subdivision is a method for creating a new set of control points C' and a new set of basis functions B', such that the new control points and basis functions define the same limit curve as the originals, yet the control points themselves are a better approximation of the limit curve. A benefit of subdivision is that one only needs to be concerned with the basis functions when deriving the subdivision filter; after that, the basis functions exist only implicitly.

The first curve subdivision scheme in graphics is credited to George Chaikin [1], dating back to 1974. In this scheme, every line segment is contracted about its centroid by a factor of one-half, and the endpoints of these contracted segments are joined to form the new curve. Equivalently, point c_i is replaced by two points on the line segment between c_i and c_{i+1}: one at $\frac{1}{4}c_i + \frac{3}{4}c_{i+1}$ and another at $\frac{3}{4}c_i + \frac{1}{4}c_{i+1}$. The subdivision mask is therefore $\left(\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4}\right)$.

In 1975, Lane and Riesenfeld [2] showed that Chaikin's "corner-cutting" sub-

division scheme actually generates quadratic (2^{nd}-degree) B-spline curves in the limit. They further showed that the subdivision mask for any d^{th}-degree B-spline curve can be derived from Pascal's triangle. For instance, the subdivision mask of Chaikin can be written as $\frac{1}{2^2}(1,3,3,1)$. In general, the subdivision mask for a degree-d B-spline curve is given by $\frac{1}{2^d} \cdot (d+2)^{th}$ *row of Pascal's triangle.*

Cubic (3^{rd}-degree) B-spline curves are particularly important in computer graphics because they are low-degree but offer C^2 continuity. The subdivision mask for cubic B-splines is $\frac{1}{8}(1,4,6,4,1)$. These mask values mean that point c_i is replaced by two points: an even point, $c_{2i} = \frac{1}{8}c_{i-1} + \frac{3}{4}c_i + \frac{1}{8}c_{i+1}$ and an odd point, $c_{2i+1} = \frac{1}{2}c_i + \frac{1}{2}c_{i+1}$.

Curve subdivision methods can be extended to higher-dimensional objects, such as surfaces, represented as a 2D grid of control vertices, or volumes, represented as a 3D lattice of control points. In these cases, the curve subdivision method is simply applied independently along the rows/columns/et cetera. Curve subdivision can also be extended, though less easily, to objects whose vertices are connected arbitrarily, i.e. polygonal meshes.

There are numerous subdivision schemes for polygonal meshes. Catmull-Clark subdivision [3] is a scheme for arbitrary polygonal meshes based on bivariate cubic B-splines. Loop subdivision [4] is a mesh subdivision scheme that operates on triangle meshes, based on three-directional quartic box splines. Another popular scheme is Doo-Sabin subdivision [5] for arbitrary polygonal meshes, which is an extension of bivariate quadratic (2^{nd}-degree) B-splines.

3.2 Multiresolution

Most research in multiresolution to date has focused on wavelets. The theory of wavelets has a rich history that spans many scientific domains, primarily engineering [6, 7]. Stollnitz et al. concisely describe wavelets as "a mathematical tool for hierarchically decomposing functions" [8]. Wavelet systems decompose a function into a coarse approximation of the function plus some details; where the coarse vertices are coefficients of the basis functions, the details are coefficients of a set of wavelet functions.

Subdivision exists within a set of nested function spaces, say S^k and S^{k+1}. These two spaces are related by their scaling functions: Φ^k form a basis for S^k, while Φ^{k+1} form a basis for S^{k+1}. The transpose $\mathbf{P}^T$ of the subdivision matrix defines the relationship between these bases.

Functions in S^{k+1} can be expressed in terms of parts in S^k and remaining parts in S^{k+1}, and for that we need a convenient basis Ψ^k for the complement

of S^k in S^{k+1}. So, the space S^{k+1} has two bases: Φ^{k+1}, the fine scale basis; and the union of Φ^k and Ψ^k, the coarse scaling functions plus the wavelet basis. A coarse function in S^k can be represented by $\sum_i c_i \Phi_i^k$, and the reconstruction of a fine function in S^{k+1} from the associated coarse function and detail information is done by $\sum_i c_i \Phi_i^k + \sum_j d_j \Psi^k$. The relationship between Φ^{k+1} and Ψ^k defines the $\mathbf{Q}$ matrix [9].

Wavelet systems are classified according to the relationship between the wavelets and the scaling functions. Stollnitz et al. [9] provide an excellent overview of wavelet classifications, which we summarize here.

Orthogonal wavelets require that "the scaling functions are orthogonal to one another, the wavelets are orthogonal to one another, and each of the wavelets is orthogonal to every coarser scaling function" [9]. In such a setting, the determination of the multiresolution filters is quite easy. Unfortunately, orthogonality is difficult to satisfy for all but the most trivial scaling functions.

Semiorthogonal wavelets relax the orthogonality conditions greatly, only requiring that each wavelet is orthogonal to all coarser scaling functions. By relaxing the constraints on the wavelets, it is easier to derive a $\mathbf{Q}$ filter (note that there is no unique choice of $\mathbf{Q}$, but there are some choices that are better than others). The drawback of semiorthogonal wavelets is that while $\mathbf{P}$ and $\mathbf{Q}$ will be sparse matrices – meaning that reconstruction can be done in linear time – the decomposition filters $\mathbf{A}$ and $\mathbf{B}$ offer no such guarantee. It often turns out that the decomposition filters are full matrices, meaning that decomposition could take quadratic time.

Finally, there are *biorthogonal* wavelets that have many of the properties of semi-orthogonal wavelets but enforce no orthogonality conditions. The only condition in a biorthogonal setting is that $[\mathbf{P}|\mathbf{Q}]$ is invertible, which by Eqn. 1 implies that the decomposition filters $\mathbf{A}$ and $\mathbf{B}$ exist (clearly this is the minimum condition we need to satisfy). Biorthogonal wavelets allow a lot of freedom in the selection of multiresolution filters, so it is usually possible to have a full set of sparse filters and therefore linear-time reconstruction *and* decomposition. Warren [10] introduced such a sparse matrix construction for determining simple biorthogonal wavelets for binary subdivision schemes.

Sweldens' *lifting* scheme [11,12] is a general method for designing biorthogonal wavelet systems. First, some initial biorthogonal filters $\mathbf{P}'$, $\mathbf{Q}'$, $\mathbf{A}'$, and $\mathbf{B}'$ are selected. Then, these initial filters are *lifted* by a matrix $\mathbf{S}$ to create a new biorthogonal system:

$$\begin{aligned} \mathbf{Q} &= \mathbf{Q}' - \mathbf{P}'\mathbf{S} \\ \mathbf{A} &= \mathbf{A}' + \mathbf{S}\mathbf{B}' \ . \end{aligned} \tag{2}$$

$\mathbf{P}'$ and $\mathbf{B}'$ are unchanged by lifting. The choice of $\mathbf{S}$ determines the properties of the wavelet system, such as the number of zero moments.

Samavati and Bartels pioneered a different multiresolution approach based on local linear conditions (LLC) [13,14]. The idea of the LLC method is to derive the $\mathbf{A}$ filter by minimizing the *local* error in the coarse approximation. Just as subdivision hides the underlying scaling functions, the LLC method does not use the scaling functions Φ^j or the wavelet functions Ψ^j directly.

Bartels and Samavati were able to apply this approach quite successfully to B-spline subdivision curves [14]. One particularly interesting property of this approach is that it can be interpreted as a semiorthogonal wavelet system with a non-standard inner product definition.

Later, Samavati et al. applied the LLC method to the reversal of Doo-Sabin subdivision [15]. They were able to derive a full set of multiresolution filters that were sparse and efficient, and that also produced a locally optimal coarse approximation of the fine data as quantified by the least squares error metric.

3.3 Multiresolution Loop

There are several existing approaches to creating multiresolution Loop surfaces. Samavati et al. [16] presented a parameterized reversal scheme for Loop subdivision as well as variants such as Butterfly subdivision [17] (variants in the sense that the face-split structure is the same, but the mask values are different). However, only the $\mathbf{A}$ matrix is derived in their work, not a full set of multiresolution filters.

Later, Hassan and Dodgson [18] presented a reversal scheme for Chaikin curves, and briefly show how to extend it to Loop-like surfaces. Like Samavati et al., the solution contains no error representation and thus is not a full multiresolution solution.

Recently, Bertram [19] and Li et al. [20] constructed fairly stable biorthogonal wavelets for Loop subdivision. Their approach is based on rewriting the subdivision rules with some additional free parameters, such that regular subdivision is unchanged but an inversion of the rules produces a multiresolution system. Each researcher pursues a different method to determine the free parameters, but each results in a large and unwieldy set of constants to handle different vertex valences. In this paper, a biorthogonal multiresolution system is constructed by our method that provides competitive performance, but our system has far fewer parameters and offers a more streamlined implementation.

4 Our Approach

Here we present a simple method that has several advantages over earlier ones. By the wavelet constraint, we can compute details at even vertices from neighboring odd vertices. This leads to an efficient data structure, because we exploit the even-odd distinction and replace even vertices with coarse approximations and odd vertices with details. Our method works for arbitrary connectivity without the need for special cases or precomputed weights, and also for boundary cases. As we will demonstrate with our example subdivision schemes, the method is theoretically sound and general enough to apply to both curves and arbitrary-topology surfaces.

When deriving multiresolution filters, we most often have a subdivision matrix $\mathbf{P}$ for which we would like to find the remaining multiresolution filters. In wavelet approaches, these filters are derived in the sequence $\mathbf{P} \rightarrow \mathbf{Q} \rightarrow \mathbf{A}, \mathbf{B}$. The reverse subdivision LLC approach derives the filters in the sequence $\mathbf{P} \rightarrow \mathbf{A} \rightarrow \mathbf{B}, \mathbf{Q}$.

A general approach to multiresolution is to construct a *trial* decomposition filter $\widetilde{\mathbf{A}}$. Lazy wavelets [12] are one way of selecting this filter based on the scaling functions. Another simplistic method is to assume an interpolating scheme and simply discard the odd vertices. However, for most mesh subdivision schemes, neither of these methods will produce a good trial coarse approximation.

For mesh editing applications, the trial decomposition may suffice for creating a multiresolution hierarchy. For other applications such as mesh compression, however, the trial filters often produce large wavelet coefficients. Thus there are methods such as lifting to enhance the trial filters.

We present a method for both specifying the trial filters *and* enhancing them that is general enough to be applied to any subdivision scheme. In our approach, we will discover the filters in order $\mathbf{P} \rightarrow \widetilde{\mathbf{Q}} \rightarrow \widetilde{\mathbf{A}} \rightarrow \mathbf{B} \rightarrow \mathbf{A}, \mathbf{Q}$. There are two distinct components:

(1) We arrive at *trial* filters $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{Q}}$ by requiring that even details can be expressed as a linear combination of the neighboring odd details:

$$d_{even} = \sum_{i \in odd} \alpha_i d_i \ . \tag{3}$$

In this way we only need to store details at the odd vertices, which ensures that the coarse data plus the details will not require more storage space than the fine data. From this constraint on the details, we can immediately get a trial approximation of the coarse data.

(2) We then use analytical optimization locally to refine our trial filters; specifically, we want to reduce the magnitude of the details. By reducing the magnitude of the details our filters will have more applicability to compression applications, because the subdivision of a coarse approximation, without details, will be closer to the original fine data.
(3) To temper the interdependence of each local optimization, we soften the refinement at each vertex and reach a more optimal solution.

To illustrate this approach, Section 5 will construct a set of multiresolution filters for cubic B-spline curve subdivision. The strength of this approach is that it is extensible to more complicated subdivision schemes including mesh schemes. Though our approach is numerical, it fits in well with previous work in wavelets. In particular, as we will see in Section 5.5, the optimization phase is equivalent to the lifting process and our approach constructs biorthogonal wavelet systems. In Section 6, we will demonstrate the power of our approach by constructing a full set of filters for Loop subdivision.

Evaluation

To objectively evaluate the quality of a multiresolution system, the least-squares error metric is used to measure the error introduced by decomposition. More precisely, if a fine mesh $\mathbf{C}^k$ (where superscript denotes level in the multiresolution hierarchy) is decomposed j times to a coarse mesh $\mathbf{C}^{k-j}$, then the error in $\mathbf{C}^{k-j}$ is quantified by the difference between $\mathbf{C}^k$ and $\mathbf{P}^j\mathbf{C}^{k-j}$ ($\mathbf{C}^{k-j}$ *subdivided* – not reconstructed – j times). Formally, the least-squares error $E(\mathbf{C}^{k-j})$ is defined as

$$E(\mathbf{C}^{k-j}) = \sqrt{|\mathbf{C}^k - \mathbf{P}^j\mathbf{C}^{k-j}|^2}\ . \tag{4}$$

If $\mathbf{P}^j\mathbf{C}^{k-j} = \mathbf{C}^k$, then $E(\mathbf{C}^{k-j}) = 0$. Note that this is error metric requires some effort at the implementation level to ensure that the vertices are ordered in a consistent fashion.

Both decomposition and reconstruction are linear in the number of vertices in the model. Different filters could imply different constants depending on the support of the filter (i.e. number of non-zero entries in a regular row or column, or the number of vertices considered by the associated mask). If a model at level k has m^k vertices, and m^{k+1} vertices at level $k+1$, then both decomposition and reconstruction have storage requirements in $O(2m^{k+1})$; vertices cannot be modified in-place, because the position of neighboring vertices depends on a vertex's original position.

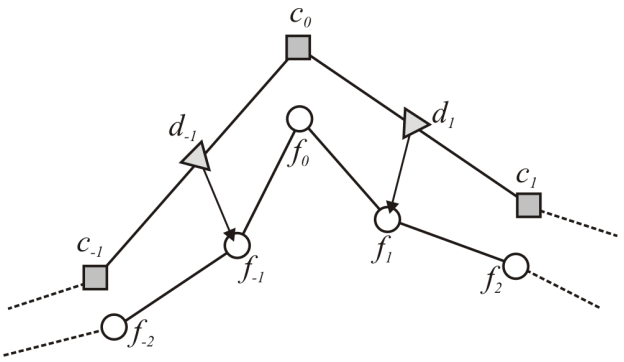

Fig. 1. Local notation for our cubic B-spline curve decomposition: a fine curve $\mathbf{F} = [\cdots f_{-1}\ f_0\ f_1\ \cdots]^T$ is decomposed to $\mathbf{C} = [\cdots c_{-1}\ c_0\ c_1\ \cdots]^T$, and odd fine points are replaced with details d_i. By the wavelet constraint, the even details can be computed from the odd details.

5 Cubic B-Spline

The goal of decomposition is to find a good coarse approximation c_0 for some fine vertex f_0. Because we want our decomposition filters to be local (i.e. banded $\mathbf{A}$ and $\mathbf{B}$ matrices), we want our coarse approximation to be based only on a local neighborhood $\ldots, f_{-1}, f_0, f_1, \ldots$ about f_0. Figure 1 shows the notation used for this local neighborhood.

5.1 Trial Filters

An important practical requirement of multiresolution filters is the storage constraint: put simply, the storage requirement for the coarse data $\mathbf{C}$ and the details $\mathbf{D}$ should not exceed that of the fine data $\mathbf{F}$. Our approach is to begin from this constraint and work outwards.

Subdivision displaces even vertices and creates odd vertices. This even-odd distinction offers a natural way to satisfy the storage requirement during decomposition: we can replace even points with coarse approximations, and odd points with details. If we do this, however, we must be able to compute the even details somehow to allow reconstruction. One possible solution is to compute an even detail d_{even} from some neighboring odd details, as in Eqn. 3.

The outcome of this constraint will be a set of trial filters, which we would like to be banded for efficiency. Therefore, we should consider only a small number of immediate neighbors of d_{even} when computing it. Additionally, because subdivision is usually a symmetric operation we can use the same coefficient

for all immediate neighbors and move the α terms outside of the summation:

$$d_{even} = \alpha \sum_i d_i,\ i \in \textit{immediate odd neighbors of } d_{even}\ .$$

According to the notation laid out in Fig. 1, the representative even vertex is d_0, neighbored by d_{-1} and d_1. Then, the above wavelet constraint becomes

$$d_0 = \alpha(d_{-1} + d_1)\ . \tag{5}$$

A detail d_i is defined as the difference between fine data and subdivided coarse data: $f_i - \widetilde{f}_i$, where the $\widetilde{f}_i$ terms result from subdivision of $\widetilde{\mathbf{C}}$. So, by the definition of cubic B-spline subdivision the details are:

$$\begin{aligned} d_0 &= f_0 - \widetilde{f}_0 = f_0 - \left(\frac{1}{8}\widetilde{c}_{-1} + \frac{3}{4}\widetilde{c}_0 + \frac{1}{8}\widetilde{c}_1\right) \\ d_{\pm 1} &= f_{\pm 1} - \widetilde{f}_{\pm 1} = f_{\pm 1} - \left(\frac{1}{2}\widetilde{c}_0 + \frac{1}{2}\widetilde{c}_{\pm 1}\right)\ . \end{aligned}$$

Because our goal is to find an expression for $\widetilde{c}_0$ that depends only on fine data $\mathbf{F}$, we should choose a value for α that eliminates all other coarse points $\widetilde{c}_{j\neq 0}$ from the expression $d_0 = \alpha(d_{-1} + d_1)$.

If we substitute the full expressions for d_{-1}, d_0, and d_1 into Eqn. 5, we can find the coefficient of $\widetilde{c}_1$ on the left side and right side. On the left side, d_0 contributes $\frac{-1}{8}\widetilde{c}_1$, while on the right side, d_1 contributes $\alpha\frac{-1}{2}\widetilde{c}_1$. To cancel these terms from the expression, we set them equal: $\frac{-1}{8} = \frac{-\alpha}{2}$. Thus $\alpha = \frac{1}{4}$, and we can write:

$$d_0 = \frac{1}{4}\left(d_{-1} + d_1\right)\ . \tag{6}$$

Equation 6 is enough to determine $\widetilde{\mathbf{Q}}$, the contribution of the details $\mathbf{D}$ to fine data $\mathbf{F}$. For even points, we find the two neighboring odd details and use α to compute the even detail according to Eqn. 6, so the row in $\widetilde{\mathbf{Q}}$ will contain $\left(\frac{1}{4}, \frac{1}{4}\right)$. For odd points, we add its stored detail, i.e. the corresponding element in $\widetilde{\mathbf{Q}}$ should be 1. Thus the reconstruction mask for $\widetilde{\mathbf{Q}}$ is $\left(\frac{1}{4}, 1, \frac{1}{4}\right)$, representing a regular column of the filter.

Because of symmetry, our choice of α will eliminate $\widetilde{c}_{-1}$ and $\widetilde{c}_1$ from Eqn. 6, leaving an expression for $\widetilde{c}_0$ in terms of fine data from F. After substituting

the full expression of each detail vector and simplifying, Eqn. 6 can be written as:

$$\widetilde{c}_0 = -\frac{1}{2} f_{-1} + 2 f_0 - \frac{1}{2} f_1 \ . \tag{7}$$

Equation 7 expresses coarse point $\widetilde{c}_i$ in terms of fine points f_{-1}, f_0, and f_1. In other words, it represents a regular row of the decomposition filter $\widetilde{\mathbf{A}}$, which can be compactly represented in mask form as $\left(-\frac{1}{2}, 2, -\frac{1}{2}\right)$. Note that this 3-element mask is the same one as arrived at, via different means, by Bartels and Samavati [14]. However, we are able to determine this trial filter without solving any bilinear systems, so our method can easily be generalized to mesh subdivision schemes.

Determining $\widetilde{\mathbf{B}}$ is straightforward now that we know $\widetilde{\mathbf{A}}$. Consider an odd detail vector d_1:

$$d_1 = f_1 - \left(\frac{1}{2}\widetilde{c}_0 + \frac{1}{2}\widetilde{c}_1\right) = \frac{1}{4} f_{-1} - f_0 + \frac{3}{2} f_1 - f_2 + \frac{1}{4} f_3 \ .$$

This represents a regular row of $\widetilde{\mathbf{B}}$, which can be expressed as $\left(\frac{1}{4}, -1, \frac{3}{2}, -1, \frac{1}{4}\right)$ in mask form. Since we only need to compute the details at odd points, $\widetilde{\mathbf{B}}$ is fully determined.

In the next section we will summarize $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{A}}$, and $\widetilde{\mathbf{B}}$ in matrix form, including a treatment of the boundary cases for open curves.

5.2 Boundary Filters

To have a complete set of multiresolution filters applicable to either open or closed curves, we should also consider the special boundary cases. For open curves, we can express each filter with a block matrix. For example, the $\mathbf{P}$ matrix would be written as: $\mathbf{P} = [\mathbf{P}_s \ \mathbf{P}_r \ \mathbf{P}_e]^T$, where $\mathbf{P}_s$ represents the filter for the start of the curve, $\mathbf{P}_r$ is applied to regular vertices, and $\mathbf{P}_e$ applies to the end of the curve. For open cubic B-spline curves, the $\mathbf{P}$ filter is:

$$\begin{bmatrix} \mathbf{P}_s \\ \mathbf{P}_r \\ \mathbf{P}_e \end{bmatrix} = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & \cdots \\ 0 & \frac{3}{4} & \frac{1}{4} & 0 & 0 & 0 & \cdots \\ 0 & \frac{3}{16} & \frac{11}{16} & \frac{1}{8} & 0 & 0 & \cdots \\ \hline 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots \\ 0 & 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} & 0 & \cdots \\ & & & \vdots & & & \\ \hline \cdots & 0 & 0 & \frac{1}{8} & \frac{11}{16} & \frac{3}{16} & 0 \\ \cdots & 0 & 0 & 0 & \frac{1}{4} & \frac{3}{4} & 0 \\ \cdots & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \cdots & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

We can apply our approach in a similar fashion as above to discover full multiresolution filters for open curves. Let f_0, f_1, f_2, and f_3 represent the four fine vertices affected by the boundary subdivision mask. Unfortunately these vertices have no clear distinction between even and odd points. For our purposes, we will consider f_0, f_1, and f_3 as even and f_2 as odd.

The best coarse approximation $\widetilde{c}_0$ for f_0 is easily found by noting that it does not move during subdivision The best coarse approximation $\widetilde{c}_1$ for f_1 is also easy to compute because f_1 is the midpoint of $\widetilde{c}_0$ and $\widetilde{c}_1$. We find $\widetilde{c}_3$ by setting $d_3 = \alpha(d_2 + d_4)$.

We do not need to compute a detail d_0 for f_0, because it will always be zero. The fact that $\widetilde{c}_0$ does not move during subdivision also allows $\widetilde{c}_1$ to be determined without error, so d_1 will also be zero. The constraint $d_3 = \alpha(d_2+d_4)$ determines d_2 and d_4.

The result of this analysis is summarized in the block matrices below.

$$\begin{bmatrix} \widetilde{\mathbf{A}}_s \\ \widetilde{\mathbf{A}}_r \\ \widetilde{\mathbf{A}}_e \end{bmatrix} = \left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ -1 & 2 & 0 & 0 & 0 & 0 & \cdots \\ \hline 0 & \frac{-1}{2} & 2 & \frac{-1}{2} & 0 & 0 & \cdots \\ & & & \vdots & & & \\ \hline \cdots & 0 & 0 & 0 & 0 & 2 & -1 \\ \cdots & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \qquad \begin{bmatrix} \widetilde{\mathbf{Q}}_s \\ \widetilde{\mathbf{Q}}_r \\ \widetilde{\mathbf{Q}}_e \end{bmatrix} = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ \hline 1 & 0 & 0 & 0 & \cdots \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & \cdots \\ & & \vdots & & \\ \hline 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{array}\right]$$

$$\begin{bmatrix} \widetilde{\mathbf{B}}_s \\ \widetilde{\mathbf{B}}_r \\ \widetilde{\mathbf{B}}_e \end{bmatrix} = \left[\begin{array}{ccccccc} \frac{3}{4} & \frac{-3}{2} & \frac{9}{8} & \frac{-1}{2} & \frac{1}{8} & 0 & \ldots \\ \hline \ldots & \frac{1}{4} & -1 & \frac{3}{2} & -1 & \frac{1}{4} & \ldots \\ & & & \vdots & & & \\ \hline \ldots & 0 & \frac{1}{8} & \frac{-1}{2} & \frac{9}{8} & \frac{-3}{2} & \frac{3}{4} \end{array}\right] .$$

5.3 Refinement

When an object $\mathbf{F}$ that is *not* the product of subdivision is decomposed with $\widetilde{\mathbf{A}}$, we find in practice that the error $E(\widetilde{\mathbf{A}}\mathbf{F})$ is large (Fig. 6(b), for instance). While there is no choice of $\mathbf{A}$ for which $E = 0$ (except trivially), some choices are better than others. In particular, having a filter with wider support (the trial mask is only 3 elements) can often provide lower-error decompositions.

We consider a strategy of displacing each coarse vertex $\widetilde{c}_i$ by some amount δ_i, with the goal of reducing the overall error in the trial coarse vertices. We therefore need to find a set of refinement vectors $\Delta = [\delta_0, \ldots, \delta_n]^T$, representing the per-vertex displacements of $\widetilde{\mathbf{C}} = [\widetilde{c_0}, \ldots, \widetilde{c_n}]^T$. The choice of Δ should satisfy $E(\widetilde{\mathbf{C}}+\Delta) < E(\widetilde{\mathbf{C}})$. This expresses the *global* error in the coarse points. However, enforcing a global reduction in error is not an extensible approach for general meshes.

Instead, we consider the *local* impact of each displacement. After refinement, representative vertex $\widetilde{c}_0$ becomes $c_0 = \widetilde{c}_0 + \delta_0$. Based on the subdivision mask of cubic B-splines, δ_0 affects the curve only in a 5-element neighborhood of fine data: $\frac{1}{8}\delta_0$ to $f_{\pm 2}$, $\frac{1}{2}\delta_0$ to $f_{\pm 1}$, and $\frac{3}{4}\delta_0$ to f_0. To have a more general method, however, we can restrict the error analysis to a smaller 3-element neighborhood.

In this case, the local error E around $\widetilde{c}_0$ can be represented by

$$E = |d_{-1}|^2 + |d_0|^2 + |d_1|^2 = |f_{-1} - \widetilde{f}_{-1}|^2 + |f_0 - \widetilde{f}_0|^2 + |f_1 - \widetilde{f}_1|^2$$

After refinement, the displacement of $\widetilde{c}_0$ by δ_0 changes the positions of $f_{\pm 1}$ (by $\frac{1}{2}\delta_0$) and f_0 (by $\frac{3}{4}\delta_0$); δ_0 should be chosen so that the new positions contain less error. The local error E_{δ_0} after refinement is expressed as

$$E_{\delta_0} = \left| f_{-1} - \left(\widetilde{f}_{-1} + \frac{1}{2}\delta_0 \right) \right|^2 + \left| f_0 - \left(\widetilde{f}_0 + \frac{3}{4}\delta_0 \right) \right|^2 + \left| f_1 - \left(\widetilde{f}_1 + \frac{1}{2}\delta_0 \right) \right|^2 ,$$

which simplifies to

$$E_{\delta_0} = a|\delta_0|^2 - \mathbf{v}^T\delta_0 + b , \tag{8}$$

where $a = \frac{17}{16}$, $\mathbf{v} = d_{-1} + \frac{3}{2}d_0 + d_1$, and $b = |d_{-1}|^2 + |d_0|^2 + |d_1|^2$.

We would like to minimize E_{δ_0}, which by the form given in Eqn. 8 is a simple quadratic optimization problem that can be solved analytically. To find the minimum of the function, we take the first derivative and set it equal to zero: $E'_{\delta_0} = 2a\delta_0 - \mathbf{v} = 0$. The solution is $\delta_0 = \frac{\mathbf{v}}{2a}$. Note that this is a minimum, because $E''_{\delta_0} = 2a > 0$. In fact, it is a unique global minimum, which follows from the fact that the Hessian $\nabla^2 E$ of E_{δ_0} is positive-definite.

We can substitute the proper values of $\mathbf{v}$ and a to get a closed-form expression for δ_0:

$$\delta_0 = \frac{\mathbf{v}}{2a} = \frac{8}{17}\left(d_{-1} + \frac{3}{2}d_0 + d_1 \right) = \frac{11}{17}\left(d_{-1} + d_1 \right) . \tag{9}$$

The final simplification step follows from Eqn. 6. Figure 2 illustrates how the refinement vector is computed; geometrically, the neighboring details give a good indication of where the original surface lies.

5.4 Partial Refinement

In our development of the refinement step, we have neglected one important aspect: the refinements are not independent. The computation of an optimal displacement for each coarse vertex assumes that all other coarse vertices will remain unchanged, i.e. will not be displaced. This is, of course, not true; every coarse point is displaced based on the positions of the trial vertices.

For an illustration of why this is problematic, consider Fig. 3. In the top left, we see a section of a fine curve and its trial decomposition, and the trial decomposition has been refined at only one vertex: $\widetilde{c}_0$. When this refined coarse curve is subdivided (bottom left), the local error is truly minimized. When *every* trial vertex is refined (top right), however, the error after subdivision

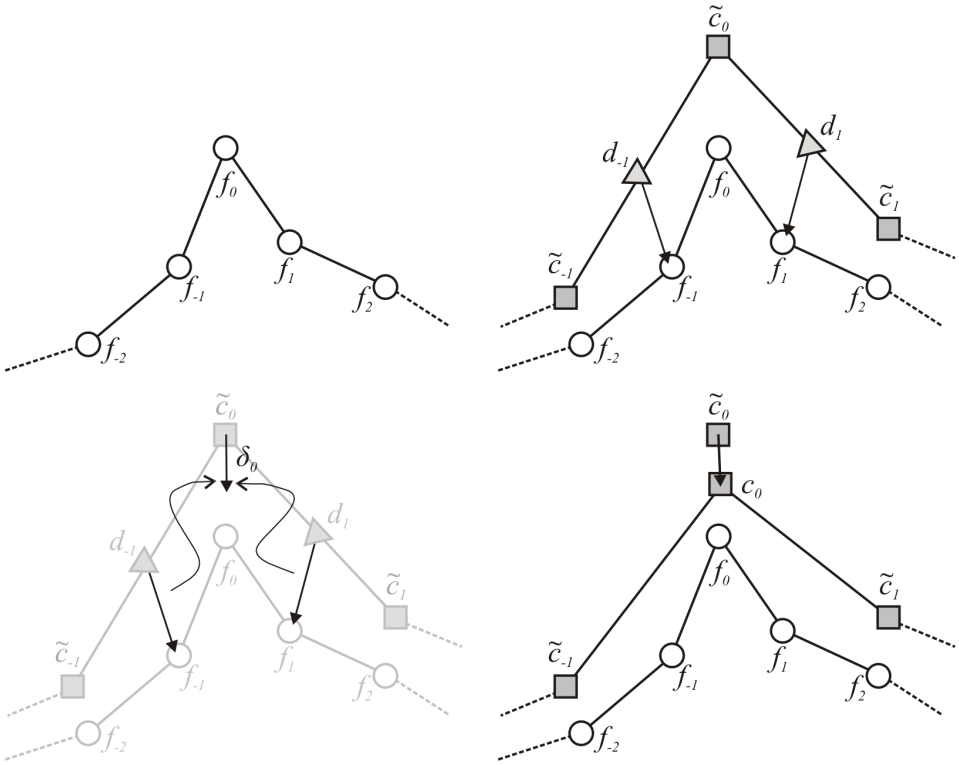

Fig. 2. Refinement of the trial vertices (top to bottom, left to right): fine data $\mathbf{F}$; coarse data $\widetilde{\mathbf{C}}$ results from applying the trial reverse filter $\widetilde{\mathbf{A}}$; δ_0 is computed from a local set of details; after refinement, the local error is reduced.

(bottom right) is no longer minimized. This is because each refinement assumes that other coarse vertices will remain in their original positions.

Intuitively, we would expect the interdependence of the displacements to cause a net "overshooting" effect, i.e. too much displacement. This is visible in Fig. 3: the subdivision of the trial points, shown in (a), lies on one side of the fine data, while the subdivision of the refined points, shown in (e), lies on the opposite side. Therefore, a partial refinement strategy is considered; instead of $c_0 = \widetilde{c}_0 + \delta_0$, each refinement should be softened by setting $c_0 = \widetilde{c}_0 + \mu\delta_0$, where $0 \leq \mu \leq 1$ is some scalar to be determined.

It is not clear how μ might be chosen, outside of trial-and-error. In general, a perfect optimization would involve a different value of μ for each coarse vertex, allowing for sensitivity to the local feature scale. However, this would lead to a decomposition mask that changes from vertex to vertex, which does not lead to a very efficient algorithm. Thus a single value for μ is a necessary simplification. An analytic approach to the selection of μ might be considered, but any local approach would still suffer from interdependence problems. Most importantly, however, it is necessary to have a method that can be extended to mesh schemes.

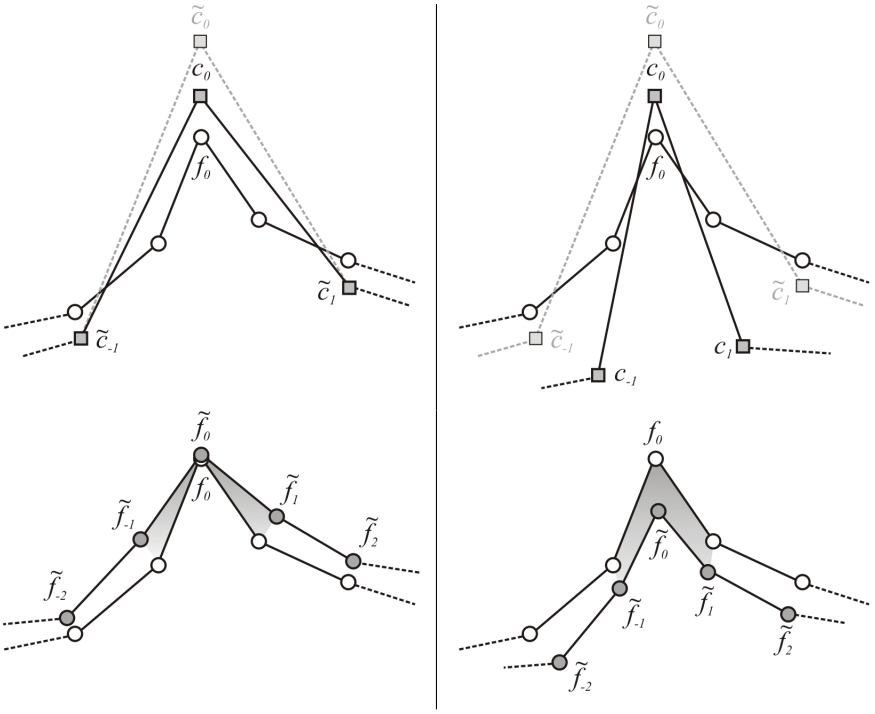

Fig. 3. Local vs. global effects of refinement. Left: when only $\widetilde{c}_0$ is displaced by refinement (top), the local error after subdivision is minimized (bottom). Right: when all coarse vertices $\widetilde{c}_i$ are displaced by refinement (top), the local error is not minimized (bottom). The shaded regions in the bottom figures indicate the local error size.

We consider a voting strategy, based on the observation that $\widetilde{c}_0$ wants to take the full refinement step, while neighbors $\widetilde{c}_{-1}$ and $\widetilde{c}_1$ want $\widetilde{c}_0$ to not move at all. In other words, the central vertex votes for $\mu = 1$, while its neighbors each vote for $\mu = 0$. However, their votes do not carry equal weight, because the position of c_0 is much more important to $\widetilde{c}_0$ than to it's neighbors. In fact, based on the subdivision filter $\mathbf{P}$, we know that $\widetilde{c}_0$ contributes $\frac{3}{4}$ to f_0, and only $\frac{1}{8}$ to $\widetilde{c}_{-1}$ and $\widetilde{c}_1$. If we adopt these weights in the voting scheme, then we have $\mu = \left(\frac{3}{4}\right) \cdot 1 + 2\left(\frac{1}{8}\right) \cdot 0 = \frac{3}{4}$. Thus $c_0 = \widetilde{c}_0 + \frac{3}{4}\delta_0$, or equivalently we can redefine δ_0 as

$$\delta_0 \leftarrow \frac{3}{4}\delta_0 = \frac{33}{68}(d_{-1} + d_1) \ . \tag{10}$$

Note that μ is equal to the central weight of the subdivision filter.

5.5 Closed-Form Filters

There is a question of where the refinement process fits into the usual multiresolution framework. One possibility is to treat the refinement process as

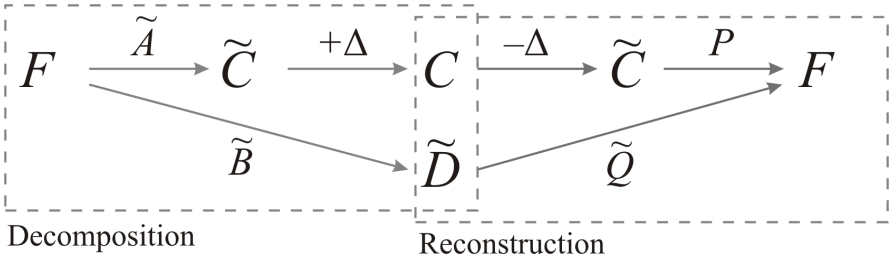

Fig. 4. The multiresolution framework with refinement.

a separate step in decomposition, which is later undone before the normal reconstruction process, as in Fig. 4.

Decomposition would now be a four-step process, where the latter two steps are for refinement:

(1) Compute the coarse approximation: $\widetilde{\mathbf{C}} = \widetilde{\mathbf{A}}\mathbf{F}$.
(2) Compute the details: $\widetilde{\mathbf{D}} = \widetilde{\mathbf{B}}\mathbf{F}$.
(3) Compute the refinement vectors, Δ.
(4) Refine the coarse approximation: $\mathbf{C} = \widetilde{\mathbf{C}} + \Delta$.

Similarly, reconstruction requires a couple of additional steps to undo the refinement:

(1) Compute the refinement vectors, Δ.
(2) Undo the refinement of the coarse approximation: $\widetilde{\mathbf{C}} = \mathbf{C} - \Delta$.
(3) Reconstruct the fine data: $\mathbf{F} = \mathbf{P}\widetilde{\mathbf{C}} + \widetilde{\mathbf{Q}}\widetilde{\mathbf{D}}$.

To summarize what we have accomplished to this point, we have established trial filters $\widetilde{\mathbf{A}}$, $\widetilde{\mathbf{B}}$, and $\widetilde{\mathbf{Q}}$ by solving the wavelet constraint (Eqn. 5). These filters were then improved by displacing each trial coarse point $\widetilde{c}_0$ by a refinement vector δ_0.

An important observation is that the refinement vectors are determined by linear combinations of elements of $\mathbf{D}$. This means that the entire set Δ of refinement vectors can be computed by some matrix $\mathbf{L}$ as $\Delta = \mathbf{LD}$. The entries of $\mathbf{L}$ are defined by the expression for δ_0 (Eqn. 10).

Using this form for Δ, we can then incorporate the refinement step into the trial filters to produce single, closed-form decomposition filters. Consider the expression $\mathbf{C} = \widetilde{\mathbf{C}} + \Delta$, which can be rewritten as

$$\mathbf{C} = \widetilde{\mathbf{A}}\mathbf{F} + \mathbf{LD} = (\widetilde{\mathbf{A}} + \mathbf{L}\widetilde{\mathbf{B}})\mathbf{F} \ ,$$

where

$$\mathbf{A} = \widetilde{\mathbf{A}} + \mathbf{L}\widetilde{\mathbf{B}} \tag{11}$$

is a closed-form decomposition filter.

The reconstruction step, $\mathbf{F} = \mathbf{P}\widetilde{\mathbf{C}} + \widetilde{\mathbf{Q}}\mathbf{D}$, can similarly be expressed in a closed form by rewriting it as

$$\mathbf{F} = \mathbf{P}(\mathbf{C} - \Delta) + \widetilde{\mathbf{Q}}\mathbf{D} = \mathbf{P}\mathbf{C} + (\widetilde{\mathbf{Q}} - \mathbf{P}\mathbf{L})\mathbf{D} \ ,$$

where

$$\mathbf{Q} = \widetilde{\mathbf{Q}} - \mathbf{P}\mathbf{L} \tag{12}$$

is our closed-form reconstruction filter. Note that it is unnecessary to alter the $\widetilde{\mathbf{B}}$ filter to reflect the refinement. Instead, the reconstruction step is altered to correctly interpret the original details.

The $\mathbf{L}$ matrix *lifts* the trial filters, just as the $\mathbf{S}$ matrix acts in Sweldens' lifting scheme (Eqn. 2). Thus the refinement step can be viewed as a way to determine the lifting matrix, and overall, our method is a way to construct biorthogonal wavelet systems.

For cubic B-spline curve refinement, the matrix $\mathbf{L}$ is defined by Eqn. 10. Note that it is unnecessary to lift the boundary points, because their details are always zero. Thus $\mathbf{L}$ is

$$\begin{bmatrix} \mathbf{L}_s \\ \mathbf{L}_r \\ \mathbf{L}_e \end{bmatrix} = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & ... \\ 0 & 0 & 0 & 0 & ... \\ \hline \frac{33}{68} & \frac{33}{68} & 0 & 0 & ... \\ 0 & \frac{33}{68} & \frac{33}{68} & 0 & ... \\ & & \vdots & & \\ \hline ... & 0 & 0 & 0 & 0 \\ ... & 0 & 0 & 0 & 0 \end{array}\right] .$$

When $\mathbf{L}$ is used to lift $\widetilde{\mathbf{Q}}$ to $\mathbf{Q} = \widetilde{\mathbf{Q}} - \mathbf{PL}$, the resulting filter is

$$\begin{bmatrix} \mathbf{Q}_s \\ \mathbf{Q}_r \\ \mathbf{Q}_e \end{bmatrix} = \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \frac{239}{272} & \frac{-33}{272} & 0 & 0 & \dots \\ \frac{-91}{1088} & \frac{-157}{1088} & \frac{-33}{544} & 0 & \dots \\ \hline \frac{-33}{68} & \frac{35}{68} & \frac{-33}{68} & 0 & \dots \\ \frac{-33}{544} & \frac{-95}{544} & \frac{-95}{544} & \frac{-33}{544} & \dots \\ & & \vdots & & \\ \hline \dots & 0 & \frac{-33}{544} & \frac{-157}{1088} & \frac{-91}{1088} \\ \dots & 0 & 0 & \frac{-33}{272} & \frac{239}{272} \\ \dots & 0 & 0 & 0 & 0 \\ \dots & 0 & 0 & 0 & 0 \end{array}\right].$$

The associated mask representation is $\left(\frac{-33}{544}\ \frac{-33}{68}\ \frac{-95}{544}\ \frac{35}{68}\ \frac{-95}{544}\ \frac{-33}{68}\ \frac{-33}{544}\right)$.

Similarly, $\mathbf{L}$ can be used to lift $\widetilde{\mathbf{A}}$ to $\mathbf{A} = \widetilde{\mathbf{A}} + \mathbf{L}\widetilde{\mathbf{B}}$, resulting in

$$\begin{bmatrix} \mathbf{A}_s \\ \mathbf{A}_r \\ \mathbf{A}_e \end{bmatrix} = \left[\begin{array}{ccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \frac{99}{272} & \frac{-99}{136} & \frac{91}{544} & \frac{173}{136} & \frac{157}{544} & \frac{-33}{68} & \frac{33}{272} & 0 & 0 & \dots \\ \hline 0 & 0 & \frac{33}{272} & \frac{-33}{68} & \frac{95}{272} & \frac{35}{34} & \frac{95}{272} & \frac{-33}{68} & \frac{33}{272} & \dots \\ & & & & \vdots & & & & & \\ \hline \dots & 0 & 0 & \frac{33}{272} & \frac{-33}{68} & \frac{157}{544} & \frac{173}{136} & \frac{91}{544} & \frac{-99}{136} & \frac{99}{272} \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

The asociated mask for the regular portion of $\mathbf{A}$ is $\left(\frac{33}{272}\ \frac{-33}{68}\ \frac{95}{272}\ \frac{35}{34}\ \frac{95}{272}\ \frac{-33}{68}\ \frac{33}{272}\right)$.

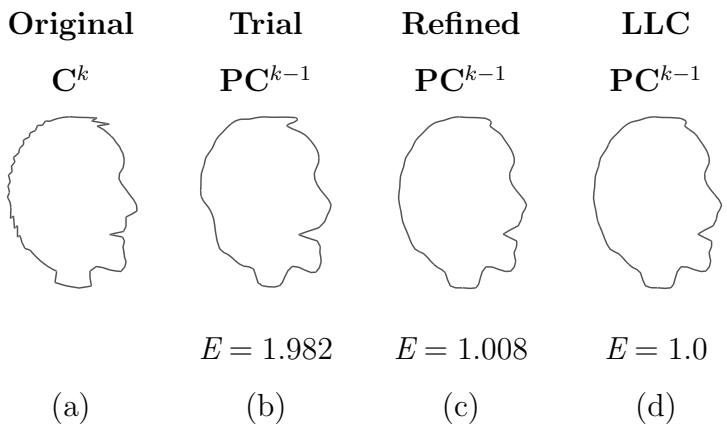

Fig. 5. A curve with 96 points (a) is decomposed once and then subdivided without details using: (b) our trial filter; (c) our refined 7-element filter; and (d) the LLC filter.

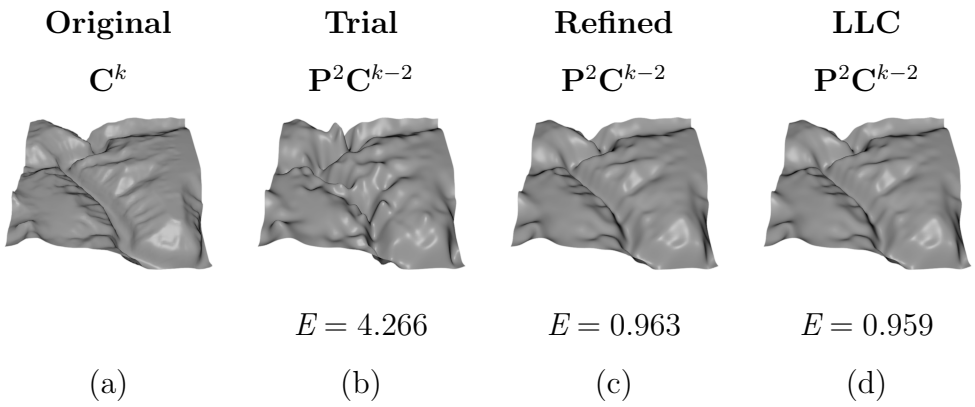

Fig. 6. A $97x97$ terrain model (a) is decomposed twice in each direction and then subdivided without details using: (b) our trial filter; (c) our refined 7-element filter; and (d) the LLC filter. (Terrain source: *http://seamless.usgs.gov*).

5.6 Results

For cubic B-splines, we have the fortunate situation of having some previously established filters with observedly good performance to compare against. Bartels and Samavati [14] present several 7-element masks with near-minimum norms. We chose the first such mask – $\left(\frac{23}{196}\ \frac{-23}{49}\ \frac{9}{28}\ \frac{52}{49}\ \frac{9}{28}\ \frac{-23}{49}\ \frac{23}{196}\right)$ – for comparison; we will refer to this mask as the LLC mask. Our refined mask is evaluated against the trial mask, $\left(\frac{-1}{2}\ 2\ \frac{-1}{2}\right)$, and the LLC mask. The support of the refined filter and the LLC filter are both seven elements, so their running times are both linear with the same constant; the trial mask has a smaller constant, but the difference is negligible for models of the size used here.

The various masks were used to decompose both curves and surfaces, which were then reconstructed without details. We then computed the error of each

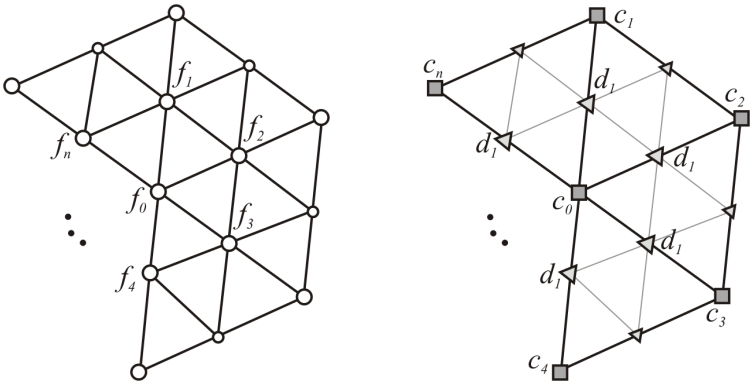

Fig. 7. Local notation for our Loop surface decomposition. During decomposition of fine data (left), a representative even vertex f_0 is replaced by coarse vertex c_0, while edge neighbors $f_1, \ldots, f_n$ are replaced with details $d_1, \ldots, d_n$ (right).

reconstructed model relative to the original model according to Eqn. 4 to get a quantitative measure of filter performance. Figures 5 and 6 show a pair of results in detail; the numerical results from these and several other models are summarized in Fig. 12.

Figure 5(a) shows a curve representing a face in profile. The curve was decomposed once and then reconstructed with (b) our trial filter, (c) our 7-element filter, and (d) the LLC filter. We can see from the errors that our refined filter is only about 1% higher than the LLC filter.

Figure 6(a) shows a section of real terrain data collected by the United States Geological Survey [21]. The terrain was decomposed with the (b) trial, (c) refined, and (d) LLC filters. Here the trial filter is utterly incapable of dealing with the complex geometry, performing over 400% worse than the LLC filter and obscuring important features like the river valley. Meanwhile, our refined filter is a mere 0.5% higher than the LLC and both are able to preserve the river valley and other features.

6 Loop Subdivision

Now that we have seen our approach applied to a simple subdivision scheme, we can extend it to a more complex surface subdivision scheme. Loop subdivision is a scheme that operates strictly on triangle meshes and has C^2 continuity at regular (valence 6) vertices. Loop subdivision is quite popular for triangle meshes because of its smoothness and also its ease of implementation.

6.1 Subdivision Filter

Loop subdivision [4] is subdivision scheme for triangle meshes; each iteration of subdivision inserts a vertex along each edge of the mesh, splitting each triangle into four. Consider a vertex c_0 of valence n, as depicted in Fig. 7. The Loop subdivision filter for even vertices yields a fine point f_0 according to the filter:

$$f_0 = (1 - n\beta)c_0 + \beta \sum_{i=1}^{n} c_i \ , \tag{13}$$

while odd vertices use the filter

$$f_i = \frac{3}{8}(c_0 + c_i) + \frac{1}{8}(c_{i-1} + c_{i+1}) \ , \tag{14}$$

where $\beta = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\left(\frac{2\pi}{n}\right)\right)^2\right)$. Note that indices of f_i and c_i should be interpreted modulo n. These two equations define the subdivision masks for Loop subdivision.

6.2 Trial Filters

Based on the wavelet constraint expressed in Eqn. 3, our goal is to determine a scalar α such that our even details can be expressed in terms of surrounding odd details. For the local Loop neighborhood, we can express the wavelet constraint as:

$$d_0 = \alpha(d_1 + d_2 + \ldots + d_n) \ , \tag{15}$$

where detail vector d_i captures the difference between fine data f_i and subdivided coarse data $\widetilde{f}_i$: $d_i = f_i - \widetilde{f}_i$. In Loop subdivision we have

$$\begin{aligned} d_0 &= f_0 - \left((1 - n\beta)\,\widetilde{c}_0 + \beta \sum_{i=1}^{n} \widetilde{c}_i\right) \ , \\ d_i &= f_i - \left(\frac{3}{8}\,(\widetilde{c}_0 + \widetilde{c}_i) + \frac{1}{8}\,(\widetilde{c}_{i-1} + \widetilde{c}_{i+1})\right) \ , \qquad i = 1, 2, \ldots, n \ . \end{aligned}$$

We determine α by substituting these full detail expressions into Eqn. 15. Because of symmetry, we only need to consider one of c_0's neighbors, say c_1. On the left side of Eqn. 15, the coefficient of c_1 is $-\beta$; on the right side, the

coefficient is $-\frac{3\alpha}{8} - 2\frac{\alpha}{8} = -\frac{5\alpha}{8}$. By setting the left- and right-side coefficients equal, c_1 (and all other neighbors of c_0) will be eliminated:

$$-\beta = -\frac{5\alpha}{8} \quad \longrightarrow \quad \alpha = \frac{8\beta}{5}$$

Thus Eqn. 15 becomes

$$d_0 = \frac{8\beta}{5} \sum_{i=1}^{n} d_i \ . \tag{16}$$

From Eqn. 16, we can immediately determine the contribution of $\mathbf{D}$ to $\mathbf{F}$, i.e. $\widetilde{\mathbf{Q}}$. For even vertices we find all of the surrounding details d_i and scale the sum by α, exactly as in Eqn. 16. For odd vertices, we simply find the corresponding detail, which is stored.

We can also determine $\widetilde{\mathbf{A}}$ from Eqn. 16. Because of symmetry, our choice of α will eliminate $\widetilde{c}_{i\neq 0}$ from the equation, leaving an expression involving only coarse vertex $\widetilde{c}_0$ and fine data f_i. After substituting the full expression of each detail vector into Eqn. 16 , we get:

$$\widetilde{c}_0 = \frac{1}{1-n\alpha} f_0 - \frac{\alpha}{1-n\alpha} \sum_{i=1}^{n} f_i \ . \tag{17}$$

Equation 17 represents a decomposition mask for the $\widetilde{\mathbf{A}}$ filter.

The role of $\widetilde{\mathbf{B}}$ is to compute the details, $\mathbf{D} = \widetilde{\mathbf{B}}\mathbf{F}$. By our wavelet constraint (Eqn. 15), we only need to compute and store details at odd vertices. Consider a representative odd detail, d_i:

$$d_i = f_i - \widetilde{f}_i = f_i - \frac{3}{8}\left(\widetilde{c}_0 + \widetilde{c}_i\right) - \frac{1}{8}\left(\widetilde{c}_{i-1} + \widetilde{c}_{i+1}\right) \ , \qquad i = 1, 2, \ldots, n \ .$$

It is feasible to find an expression for d_i that depends only on fine data f_i by replacing each $\widetilde{c}_i$ according to Eqn. 17, but that requires enumerating the 1-ring of four different vertices and determining where they overlap. It is more practical to simply use the form above to compute the odd details.

6.3 Refinement

Again we wish to find a refinement vector δ_0 that can reduce the local error about $\widetilde{c}_0$. As before, we consider the 1-ring of neighbors of $\widetilde{c}_0$; if we displace $\widetilde{c}_0$

by δ_0, how does the error change? Let E_{δ_0} be the error after refinement. Then

$$E_{\delta_0} = |d_0 - (1 - n\beta)\,\delta_0|^2 + \left|d_1 - \frac{3}{8}\delta_0\right|^2 + \ldots + \left|d_n - \frac{3}{8}\delta_0\right|^2 \tag{18}$$

based on the contribution of $(1 - n\beta)\delta_0$ to $\widetilde{f_0}$ and $\frac{3}{8}\delta_0$ to $\widetilde{f_1}, \ldots, \widetilde{f_n}$.

To reduce the error in our trial filter, we should choose δ_0 such that E_{δ_0} is minimized. Simplifying Eqn. 18 yields a form that can again be solved analytically:

$$E_{\delta_0} = a|\delta_0|^2 - \mathbf{v}^T\delta_0 + b \ ,$$

where

$$\begin{aligned} a &= n\frac{9}{64} + (1 - n\beta)^2 \ , \\ \mathbf{v} &= 2\,(1 - n\beta)\,d_0 + \frac{3}{4}\sum_{i=1}^{n} d_i \ , \\ b &= \sum_{i=0}^{n} |d_i|^2 \ . \end{aligned}$$

This is of the same form as the cubic B-spline error function from Eqn. 8. Setting $\delta_0 = \mathbf{v}/2a$, we find:

$$\begin{aligned} \delta_0 &= \frac{\mathbf{v}}{2a} \\ &= \frac{2\,(1 - n\beta)\,d_0 + \frac{3}{4}\sum_{i=1}^{n} d_i}{2\left(\frac{9n}{64} + (1 - n\beta)^2\right)} \\ &= \frac{2\,(1 - n\beta)\,\frac{8\beta}{5}\sum_{i=1}^{n} d_i + \frac{3}{4}\sum_{i=1}^{n} d_i}{2\left(\frac{9n}{64} + (1 - n\beta)^2\right)} \\ \delta_0 &= \kappa\sum_{i=1}^{n} d_i \ , \end{aligned} \tag{19}$$

where Eqn. 16 is used to eliminate d_0 and

$$\kappa = \frac{1}{(1 - n\beta)^2 + \frac{9}{32}n}\left[\frac{8}{5}\beta(1 - n\beta) + \frac{3}{8}\right] \ .$$

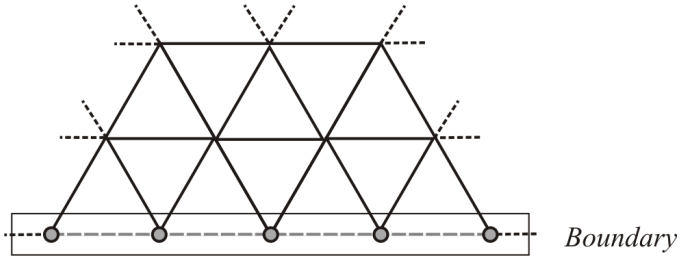

Fig. 8. Boundary vertices and edges in a Loop model can be decomposed with cubic B-spline filters.

6.4 Partial Refinement

Using the voting strategy, $\widetilde{c}_0$ again wants to take the full refinement step, while neighbors $\widetilde{c}_i$ want $\widetilde{c}_0$ to not move at all. The central vertex votes for $\mu = 1$ with a weight of $1 - n\beta$, while the first-ring neighbors vote for $\mu = 0$ with a weight of β. Thus

$$\mu = (1 - n\beta)\, 1 + (n\beta)\, 0 = 1 - n\beta \ . \tag{20}$$

Therefore

$$\kappa = \frac{(1 - n\beta)}{(1 - n\beta)^2 + \frac{9}{32}n} \left[\frac{8}{5}\beta(1 - n\beta) + \frac{3}{8} \right] \ .$$

6.5 Boundary Filters

A subdivision mask can only be applied when a complete neighborhood exists. For interior vertices, a full neighborhood is always defined, as well as for interior edges. If there is a boundary in a mesh, however, then vertices and edges that make up the boundary do not have full neighborhoods. Special boundary masks must be defined to handle such cases. See Fig. 8.

According to the Loop subdivision rules, boundary vertices are subdivided according to cubic B-spline subdivision. Thus for multiresolution boundary filters, a cubic B-spline multiresolution should be used, such as the systems described in Sec. 5.5. In particular, the regular filters $\mathbf{A}_r$, $\mathbf{B}_r$, and $\mathbf{Q}_r$ are used along any continuous boundary, while the boundary cases $\mathbf{A}_{s/e}$, $\mathbf{B}_{s/e}$, and $\mathbf{Q}_{s/e}$ can be applied to corner points.

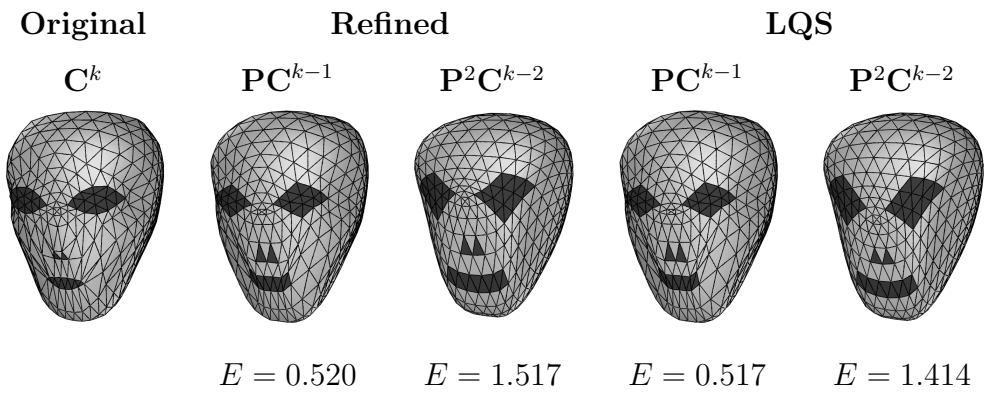

Fig. 9. An alien model consisting of 1536 faces (left) is decomposed once and twice with our refined filter, and then with the LQS filter. The decomposed model is then subdivided to the original resolution for error measurement.

6.6 Closed Form

As with the simple curve case, the refinement vector to improve our trial Loop decomposition filter is a linear combination of the odd details surrounding our representative vertex. So, if $\mathbf{D}$ is a vector of all our detail vectors, we can write all refinement vectors as a vector Δ such that $\Delta = \mathbf{LD}$ for some matrix $\mathbf{L}$, where $\mathbf{L}$ is determined by the refinement stage (Eqn. 19).

A side-effect of refinement is a widening of the decomposition mask. We saw it in the cubic B-spline case, where our trial filter of width 3 became a filter of width 7. This widening arises because we are incorporating information from surrounding details, which in turn are based on a wider neighborhood of fine vertices.

In fact, a closed-form of our Loop decomposition filter after refinement would require a 3-ring of neighbors about the central vertex. Even if all vertices in the mesh are regular (valence-6), enumerating a vertex's 3-ring is a challenging problem.; in a non-regular setting, enumerating a 3-ring is even more difficult. So, it seems that for Loop surfaces we must be content with a step-wise decomposition as depicted in Fig. 4.

6.7 Results

For evaluation of our Loop filters, we compare our filter's performance with the multiresolution Loop scheme of Li et al. [20], which we will refer to as the LQS filter. The performance of each filter is measured according to Eqn. 4. The LQS filter has slightly wider support than the refined filter, but the difference in running time between the two filters is not noticeable.

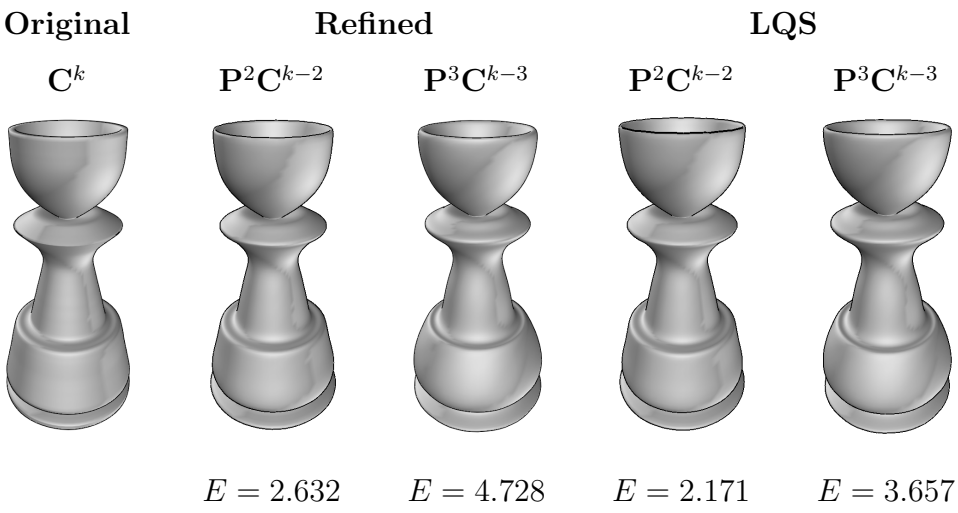

Fig. 10. A pawn model consisting of 19456 faces (left) is decomposed once and twice with our refined filter, and then with the LQS filter. The decomposed model is then subdivided to the original resolution for error measurement.

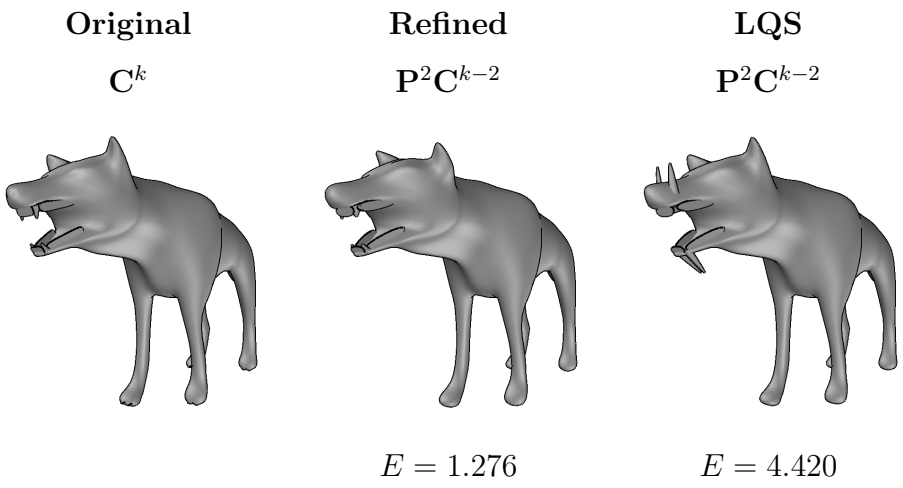

Fig. 11. A wolf model consisting of 9056 faces is decomposed twice with the refined filter and the LQS filter, and then subdivided to the original resolution.

Figure 9 depicts a model of an alien head (left). This model has relatively few sharp edges, but contains a lot of tangential vertex displacement. The model is decomposed once and twice with both the refined filter and the LQS filter. Visually, the results of each filter are about the same, and the error measurements confirm the similarity of performance.

The pawn model of Fig. 10 contains some high-frequency features that put stress on a decomposition filter. The original model contains many sharp edges and flat regions that would normally be smoothed out by subdivision. After two levels of decomposition, both the refined filter and the LQS filter are able to preserve the features of the model. After the third level, both filters begin to lose some details, such as in the ridge at the top and smooth edges of the

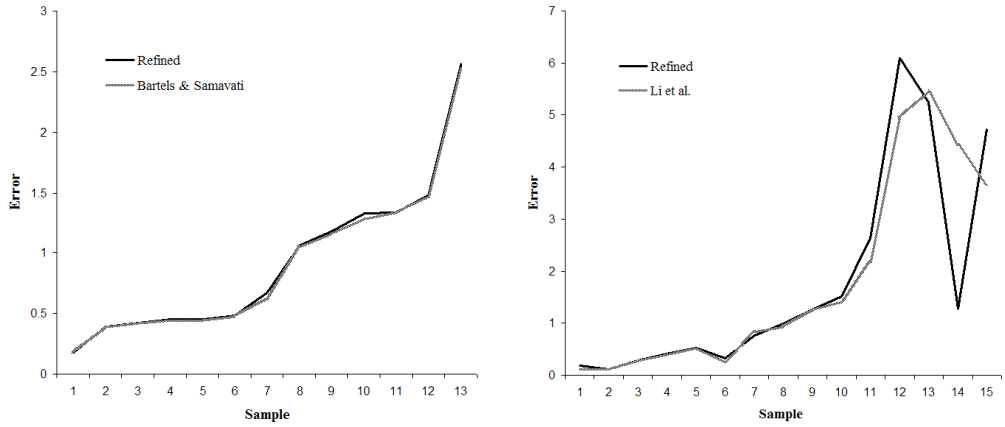

Fig. 12. Summary of error measurements from several models. *Left*: cubic B-spline results comparing our refined filter with the Bartels-Samavati filter. *Right*: Loop results for our refine filter compared against the LQS filter.

base. Visually and numerically, the filters again perform quite similarly.

Finally, Fig. 11 shows a wolf model with sharp teeth. Such features would normally be smoothed out considerably by subdivision. When the model is decomposed twice with our refined filter, the teeth are preserved as well as could be expected. The LQS filter, however, has difficulty with the feature and overcompensates, causing the teeth to protrude through the wolf's mouth. In this example, our refined filter performs considerably better.

7 Conclusion & Future Work

We have presented a general approach for constructing multiresolution filters for a broad range of subdivision schemes. Our approach exploits the distinction between even and odd vertices to find a good set of trial filters, similar to lazy wavelet constructions. The trial filters are then refined by an optimization step that seeks to reduce the error introduced during the decomposition stage.

To explore the suitability of our approach, we have applied it to cubic B-spline curves and Loop subdivision. Using our approach, we are able to quickly and easily find a set of trial filters. Our trial $\widehat{\mathbf{A}}$ filters consider a very small neighborhood (only the first-level neighbors of each even vertex) and were therefore sub-optimal in terms of the magnitude of the details.

By performing a local optimization, and then softening the results of the optimization to account for the interdependence of the local settings, we were able to vastly reduce the error in our coarse approximations. The results for both cubic B-spline curves and Loop surfaces (Fig. 12) showed that our method

performs in line with earlier methods, while being easier to understand and implement. In addition, our method allows for easy treatment of boundary cases.

Our technique is a natural fit for many subdivision schemes because it is based only on the distinction between even and odd vertices. The current direction of this research is applying it to Catmull-Clark subdivision. Loop and Catmull-Clark subdivision represent the most popular subdivision schemes, and unlike Loop, there aren't any existing multiresolution filters for Catmull-Clark.

References

[1] G. Chaikin, An Algorithm for High Speed Curve Generation, Computer Graphics and Image Processing 3 (4) (1974) 346–349.

[2] J. Lane, R. Riesenfeld, A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 2 (1) (1975) 35–46.

[3] E. Catmull, J. Clark, Recursively Generated B-spline Surfaces on Arbitrary Topological Surfaces, Computer-Aided Design 10 (6) (1978) 350–355.

[4] C. Loop, Smooth Subdivision Surfaces Based on Triangles, Master's thesis, Department of Mathematics, University of Utah (1987).

[5] D. Doo, M. Sabin, Behaviour of Recursive Subdivision Surfaces Near Extraordinary Points, Computer-Aided Design 10 (6) (1978) 356–260.

[6] I. Daubechies, Ten Lectures on Wavelets, Society for Industrial and Applied Mathematics, 1992.

[7] G. Strang, T. Nguyen, Wavelets and Filter Banks, Wellesley College, 1996.

[8] E. Stollnitz, T. DeRose, D. Salesin, Wavelets for Computer Graphics: A Primer, Part 1, IEEE Computer Graphics and Applications 15 (3) (1995) 76–84.

[9] E. Stollnitz, T. DeRose, D. Salesin, Wavelets for Computer Graphics: Theory and Applications, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.

[10] J. Warren, Sparese Filter Banks for Binary Subdivision Schemes, Mathematics of Surfaces VII.

[11] W. Sweldens, P. Schröder, Building Your Own Wavelets At Home, in: Wavelets in Computer Graphics, ACM SIGGRAPH Course notes, 1996, pp. 15–87.

[12] W. Sweldens, The Lifting Scheme: A Construction of Second Generation Wavelets, SIAM J. Math. Anal. 29 (2) (1997) 511–546.

[13] F. Samavati, R. Bartels, Multiresolution Curve and Surface Representation by Reversing Subdivision Rules, Computer Graphics Forum 18 (2) (1999) 97–120.

[14] R. Bartels, F. Samavati, Reversing Subdivision Rules: Local Linear Conditions and Observations on Inner Products, Journal of Computational and Applied Mathematics 119 (2000) 29–67.

[15] F. Samavati, N. Mahdavi-Amiri, R. Bartels, Multiresolution Surfaces having Arbitrary Topologies by a Reverse Doo Subdivision Method, Computer Graphics Forum 21 (2) (2002) 121–136.

[16] F. Samavati, H. Pakdel, C. Smith, P. Prusinkiewicz, Reverse Loop Subdivision, Tech. rep., University of Calgary, available at *http://pharos.cpsc.ucalgary.ca/* (2003).

[17] N. Dyn, D. Levin, J. Gregory, A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control, ACM Trans. Graph. 9 (2).

[18] M. F. Hassan, N. A. Dodgson, Reverse subdivision, in: N. Dodgson, M. Floater, M. Sabin (Eds.), Advances in Multiresolution for Geometric Modelling, Springer, 2005, pp. 271–283.

[19] M. Bertram, Biorthogonal Loop-Subdivision Wavelets, Computing 72 (1-2) (2004) 29–39.

[20] D. Li, K. Qin, H. Sun, Unlifted Loop Subdivision Wavelets, in: 12th Pacific Conference on Computer Graphics and Applications, 2004.

[21] USGS, US Geological Survey's Seamless Data Distribution System, available at *http://seamless.usgs.gov/website/seamless* (Oct. 2004).