Spherical Wavelets: Texture Processing

Peter Schröder[*†] and Wim Sweldens[*‡]

[*] Department of Mathematics [†] Department of Computer Science
University of South Carolina, USA

[‡] Department of Computer Science
Katholieke Universiteit Leuven, Belgium

Abstract: Wavelets are a powerful tool for planar image processing. The resulting algorithms are straightforward, fast, and efficient. With the recently developed spherical wavelets this framework can be transposed to spherical textures. We describe a class of processing operators which are diagonal in the wavelet basis and which can be used for smoothing and enhancement. Since the wavelets (filters) are local in space and frequency, complex localized constraints and spatially varying characteristics can be incorporated easily. Examples from environment mapping and the manipulation of topography/bathymetry data are given.

1 Introduction

Over the last few years, wavelets have proven themselves as a versatile tool for (planar) image processing [13]. They have been used for applications such as image compression [4], image enhancement, feature detection [12], and noise removal [5]. Wavelets are computationally attractive as the associated transform is linear in the number of pixels. In the signal processing context the transform is often referred to as subband filtering. The resulting coefficients describe the features of the underlying image in a *local* fashion in both frequency and space. Once in the wavelet domain, operators such as smoothing, enhancement, edge finding, and noise removal can be performed in a straightforward fashion.

Recently, the authors introduced a construction of wavelets on the sphere [15]. Having a fast wavelet transform on the sphere and a family of wavelets with various properties to choose from, the present paper considers their use in the processing of spherical texture maps.

Examples of such textures in computer graphics include environment maps[1] and textures on spheres. While the former are defined over the set of directions, which forms the sphere S^2 in R^3, the latter applies to the sphere as a geometric modeling primitive. In this paper we consider examples of texture processing from both of these domains, and show how spherical wavelets can be applied to these tasks.

We begin with an introduction to spherical wavelets including a description of the resulting transform and its implementation. This is followed by a discussion of diagonal operators in the wavelet domain and how they relate to the common tasks of smoothing and sharpening of images. We then give concrete examples of processing environment maps and a spherical texture subject to complex localized constraints. The paper concludes with a discussion and outlook to further applications.

[1] Thanks to Paul Haeberli for suggesting this application domain.

2 Spherical Wavelets

In this section we first give the basic subdivision of a sphere which induces the hierarchy needed as a foundation for multiresolution analysis. Next we explain the construction of spherical wavelets. We limit ourselves to the construction of bases arising from interpolating subdivision schemes. Other constructions are described in [15]. We present the fast spherical wavelet transform and its implementation.

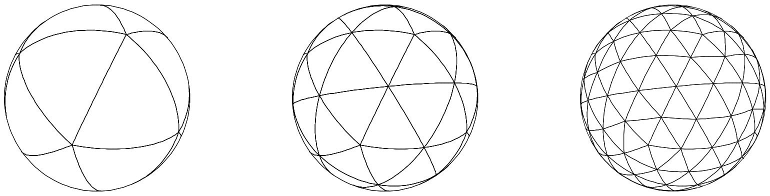

Figure 1 The geodesic sphere construction starting with the icosahedron on the left (subdivision level 0) and the next 2 subdivision levels. Successive levels are generated cutting each triangle into four children. This is accomplished by adding vertices at the midpoint of edges and connecting them with geodetics.

2.1 Subdividing a Sphere

The construction of spherical wavelets in [15] relies on a recursive partitioning of the sphere into spherical triangles. This is done starting from a Platonic solid of triangles and recursive subdivision of the triangles (see Figure 1) and is known as the geodesic sphere construction [6].

Denote the set of all vertices obtained after j subdivisions with $S_j = \{\mathbf{s}_{j,k} \in \mathrm{S}^2 \mid k \in K(j)\}$, where $K(j)$ is an index set. The vertices of the original Platonic solid are in S_0; S_1 contains those vertices and all new vertices on the edge midpoints (see Figure 1 left and middle). Since $S_j \subset S_{j+1}$ we also let $K(j) \subset K(j+1)$, so that $\mathbf{s}_{j,k} = \mathbf{s}_{j+1,k}$. The index k of a vertex in S_0 is retained when indexing the same vertex as a member of S_1 and so on. All new members of S_1, the edge midpoints, get indices m different from the ones already taken by the vertices inherited from S_0. Let $M(j) = K(j+1) \setminus K(j)$ be the indices of the vertices added when going from level j to $j+1$. The vertices on level $j+1$ thus consist of two groups: the "old" ones inherited from S_j ($\mathbf{s}_{j+1,k}$ with $k \in K(j)$) and the "new" ones ($\mathbf{s}_{j+1,m}$ with $m \in M(j)$). Indices will be used consistently in the sense that $k \in K(j)$, $l \in K(j+1)$, and $m \in M(j)$.

Figure 1 shows this subdivision scheme for several levels, beginning with the icosahedron. Choosing the icosahedron as the starting point, the resulting triangulation has the least imbalance in area between its constituent triangles. Such imbalances, most pronounced in the subdivision starting from the tetrahedron, can lead to visible artifacts. Here we will consider only the icosahedral geodesic subdivision for which $\#K(j) = 10 \cdot 4^j + 2$.

2.2 Multiresolution Analysis

We begin by defining the notion of multiresolution analysis on the sphere S^2. Consider the function space $\mathrm{L}_2 = \mathrm{L}_2(\mathrm{S}^2)$ with the area measure $d\omega$, i.e., all functions of finite energy. We define a multiresolution analysis as a sequence of closed subspaces $V_j \subset \mathrm{L}_2$, with $j \geqslant 0$, so that

1. $V_j \subset V_{j+1}$,
2. $\bigcup_{j=0}^{\infty} V_j$ is dense in L_2,
3. for each j, scaling functions $\varphi_{j,k}$ with $k \in K(j)$ exist so that $\{\varphi_{j,k} \mid k \in K(j)\}$ is a Riesz basis[2] of V_j.

In this paper we only consider interpolating scaling functions, i.e., scaling functions for which

$$\varphi_{j,k}(\mathbf{s}_{j,k'}) = \delta_{k-k'} \quad \text{for} \quad k,\, k' \in K(j)\,.$$

This can be visualized as a function centered around a given vertex where its value is 1 while it is 0 at all other vertices. Since $\varphi_{j,k} \in V_j \subset V_{j+1}$ we can write $\varphi_{j,k}$ in the basis of V_{j+1}. Because of the interpolation property this linear combination, also known as the *refinement relation*, takes the form

$$\varphi_{j,k} = \varphi_{j+1,k} + \textstyle\sum_{m \in M(j)} h_{j,k,m}\, \varphi_{j+1,m}\,. \tag{1}$$

Note that the coefficients $h_{j,k,m}$ of this linear combination can be different for every $k \in K(j)$ at a given level $j \geqslant 0$. The index m in the sum ranges over bases at new vertices $M(j) = K(j+1) \setminus K(j)$.

The easiest example of interpolating scaling functions are the hat functions; a hat function $\varphi_{j,k}$ is 1 at $\mathbf{s}_{j,k}$ and dies of linearly towards its immediate neighbors.

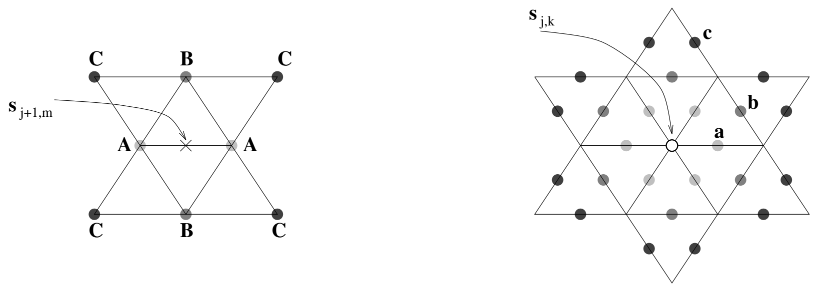

Figure 2 Local neighborhoods for the Butterfly scheme. On the left are the vertex index sets A, B, and C, whose values are used in determining the new value at the central edge midpoint. On the right are the vertex index sets a, b, and c, of all new edge midpoints to which the value of the central "old" vertex makes a contribution.

2.3 Subdivision Schemes

Interpolating scaling functions can be constructed using interpolating subdivision schemes. The idea of an interpolating subdivision scheme is the following: given the "old"

[2] A Riesz basis of a Hilbert space is a countable subset $\{f_k\}$ so that every element f of the space can be written uniquely as $f = \sum_k c_k\, f_k$, and positive constants A and B exist with $A\,\|f\|^2 \leqslant \sum_k |c_k|^2 \leqslant B\,\|f\|^2$.

values $\lambda_{j,k}$ of a function on S_j, find the "new" values on $T_j = S_{j+1} \backslash S_j$ (while preserving the values at the vertices of S_j). A simple example of this is a linear subdivision, which assigns each new edge midpoint the average of the values at the end points of its parent edge. To make matters concrete we define the following two neighborhoods:

- $n(j,k) \subset M(j)$: indices of the new values affected by the old value $\lambda_{j,k}$,
- $N(j,m) \subset K(j)$: indices of the old values who determine a new value $\lambda_{j+1,m}$.

These two sets are related through $n(j,k) = \{m \in M(j) \mid k \in N(j,m)\}$. The subdivision scheme can now be written as

$$\lambda_{j+1,m} = \sum_{k \in N(j,m)} h_{j,k,m} \, \lambda_{j,k} \, .$$

The scaling function $\varphi_{j,k}$ itself can be generated by starting on level j, setting the value $\lambda_{j,k'}$ of all vertices $\mathbf{s}_{j,k'}$ to zero except for $\mathbf{s}_{j,k}$ whose value $\lambda_{j,k}$ is set to 1, followed by an application of the subdivision scheme ad infinitum. The resulting scaling function satisfies $\varphi_{j,k}(\mathbf{s}_{j+i,l}) = \lambda_{j+i,l}$, and the following refinement relation:

$$\varphi_{j,k} = \varphi_{j+1,k} + \sum_{m \in n(j,k)} h_{j,k,m} \, \varphi_{j+1,m} \, . \tag{2}$$

Note that the set $N(j,m)$ defines the summation over the *coefficients* λ while the set $n(j,k)$ defines the summation over the *functions* φ. We may think of this as defining an adjoint relationship between N and n: $N(j,m) = n^T(j,k)$. This property is useful when changing the order of quantifiers since $[\forall k \in K(j) \ : \ \forall m \in n(j,k)]$ is equivalent to $[\forall m \in M(j) \ : \ \forall k \in N(j,m)]$.

Let us discuss two examples of interpolating subdivision schemes. Define for a vertex $\mathbf{s}_{j+1,m}$ the local neighborhoods $A(j,m)$, $B(j,m)$ and $C(j,m)$, each of them subsets of $K(j)$, and for a vertex $\mathbf{s}_{j,k}$ the neighborhoods $a(j,k)$, $b(j,k)$ and $c(j,k)$, each of them subsets of $M(j)$ as in Figure 2. The upper and lower case notations for the neighborhoods are each others adjoints (like n and N). The uppercase sets have the property that they are easier to implement than the lowercase ones. The chief reason being that they are smaller and have a fixed size ($\#A = 2$, $\#B = 2$, $\#C = 4$), while the size of the lowercase ones depends on the connectivity of $\mathbf{s}_{j,k}$. The latter becomes particularly relevant around the vertices of the original icosahedron, whose valence is 5 as opposed to 6 for all other vertices.

In a linear scheme $\lambda_{j+1,m}$ gets as value the average of the values of the two neighboring vertices in S_j.

$$\lambda_{j+1,m} = 1/2 \sum_{k \in A(j,m)} \lambda_{j,k} \, .$$

This results in the hat function referred to earlier. In the Butterfly scheme [7] a new value is found as:

$$\lambda_{j+1,m} = 1/2 \sum_{k \in A(j,m)} \lambda_{j,k} + 1/8 \sum_{k \in B(j,m)} \lambda_{j,k} - 1/16 \sum_{k \in C(j,m)} \lambda_{j,k} \, .$$

The hat functions are continuous, but not differentiable. The Butterfly scheme can lead to differentiable functions if a smooth map from the sphere to a regular planar triangulation exists. This map is determined by the partitioning of the sphere. The geodesic sphere construction presented here does not everywhere lead to such a smooth map. However, visually the basis functions are quite smooth (see the left image in row 3 of the color plate 5 for the corresponding wavelet). It is possible to build more elaborate partitionings of the sphere which ensure the differentiability of the basis functions.

2.4 Wavelets

The basic idea of wavelets is that they form a basis for a space complementing V_j in V_{j+1}. Such a space is denoted W_j so that $V_{j+1} = V_j \oplus W_j$. Note that we do not require W_j to be orthogonal to V_j as in the classical wavelet case. A basis of W_j is given by wavelet functions $\{\psi_{j,m} \mid m \in M(j)\}$ and we get

$$\mathrm{L}_2 = V_0 \oplus \bigoplus_{j=0}^{\infty} W_j\,,$$

thus constructing a basis for L_2 with

$$\{\varphi_{0,k} \mid k \in K(0)\} \cup \{\psi_{j,m} \mid j \geqslant 0,\, m \in M(j)\}\,.$$

A very simple construction for a complement space is given by $\psi_{j,m} = \varphi_{j+1,m}$ for $m \in M(j)$. However, this does not lead to a stable (Riesz) basis for L_2. Part of the problem is that the wavelets do not have a vanishing integral. Instead we propose wavelets of the form

$$\psi_{j,m} = \varphi_{j+1,m} - \textstyle\sum_{k \in A(j,m)} s_{j,k,m}\, \varphi_{j,k}. \tag{3}$$

In words, we define the wavelet at the midpoint of an edge as a linear combination of the scaling function at the midpoint $(j+1, m)$ and two scaling functions on the *coarser level* at the two endpoints of the parent edge $(j, k_{1,2})$. The weights $s_{j,k,m}$ are chosen so that the resulting wavelet has a vanishing integral

$$s_{j,k,m} = I_{j+1,m}/2\, I_{j,k} \quad \text{with} \quad I_{j,k} = \textstyle\int \varphi_{j,k}\, d\omega\,.$$

This construction of wavelets is a particular instance of a more general scheme called the "lifting scheme" [18, 17]. To facilitate notation we define for the coarsest level $M(-1) := K(0)$, $\psi_{-1,k} := \varphi_{0,k}$, and $\gamma_{-1,k} := \lambda_{0,k}$. With this notation a function $f \in \mathrm{L}_2$ can be realized as

$$f = \textstyle\sum_{-1 \leqslant j} \sum_{m \in M(j)} \gamma_{j,m}\, \psi_{j,m}\,.$$

2.5 Fast Wavelet Transform

Consider a function $f \in V_{j+1}$ given by its scaling function coefficients:

$$f = \textstyle\sum_{l \in K(j+1)} \lambda_{j+1,l}\, \varphi_{j+1,l}\,. \tag{4}$$

Note that because of the interpolating property it holds that $\lambda_{j+1,l} = f(\mathbf{s}_{j+1,l})$. We can also write this function as:

$$f = \textstyle\sum_{k \in K(j)} \lambda_{j+1,k}\, \varphi_{j,k} + \sum_{m \in M(j)} \gamma_{j,m}\, \varphi_{j+1,m}\,. \tag{5}$$

Evaluating both expressions at $\mathbf{s}_{j,k}$ confirms that we use the same $\lambda_{j+1,k}$ coefficients. The coefficients $\gamma_{j,m}$ can be found by evaluating (4) and (5) at a vertex $\mathbf{s}_{j+1,m}$:

$$\lambda_{j+1,m} = \gamma_{j,m} + \textstyle\sum_k \lambda_{j+1,k}\, \varphi_{j,k}(\mathbf{s}_{j+1,m}) = \gamma_{j,m} + \sum_{k \in N(j,m)} \lambda_{j+1,k}\, h_{j,k,m}\,. \tag{6}$$

The latter follows from the refinement relation (2). We can also write this function as

$$f = \textstyle\sum_{k \in K(j)} \lambda_{j,k}\, \varphi_{j,k} + \sum_{m \in M(j)} \gamma_{j,m}\, \psi_{j,m}\,. \tag{7}$$

This can be seen by substituting the lifting equation (3) in (7) and identifying components with (5). This results in:

$$\lambda_{j+1,k} = \lambda_{j,k} - \textstyle\sum_{m \in a(j,k)} s_{j,k,m}\, \gamma_{j,m} \;. \tag{8}$$

One step in the analysis (decomposition) goes from the $\{\lambda_{j+1,l}\}$ on one hand to the $\{\lambda_{j,k}\}$ and $\{\gamma_{j,k}\}$ on the other hand, while one step in the synthesis (reconstruction) does exactly the opposite. A complete transform from some finest level $j = n$ to $j = 0$ now follows from recursing on j.

The implementation of the `Synthesis` and `Analysis` is now straightforward and consists of two phases. Note that it only uses the the neighborhoods A, B, and C, which are the easier ones to implement.

`Analysis_Stage_I`: Calculate the $\gamma_{j,m}$ as follows (using (6)):

$$\begin{aligned}\forall m \in M(j) \,:\, \gamma_{j,m} \;:=\; & \lambda_{j+1,m} - 1/2 \textstyle\sum_{k \in A(j,m)} \lambda_{j+1,k} - \\ & 1/8 \textstyle\sum_{k \in B(j,m)} \lambda_{j+1,k} + 1/16 \textstyle\sum_{k \in C(j,m)} \lambda_{j+1,k} \;.\end{aligned}$$

`Analysis_Stage_II`: Calculate the $\lambda_{j,k}$ using the $\gamma_{j,m}$ from Stage I (using (8)):

$$\forall k \in K(j) \,:\, \lambda_{j,k} = \lambda_{j+1,k}$$

$$\forall m \in M(j) \,:\, \forall k \in A(j,m) \,:\, \lambda_{j,k} \mathrel{+}= s_{j,k,m}\, \gamma_{j,m} \;.$$

`Synthesis_Stage_I`: Calculate the $\lambda_{j+1,k}$:

$$\forall k \in K(j) \,:\, \lambda_{j+1,k} = \lambda_{j,k}$$

$$\forall m \in M(j) \,:\, \forall k \in A(j,m) \,:\, \lambda_{j+1,k} \mathrel{-}= s_{j,k,m}\, \gamma_{j,m} \;.$$

`Synthesis_Stage_II`: Calculate the $\lambda_{j+1,m}$ using the $\lambda_{j+1,k}$ from Stage I:

$$\begin{aligned}\forall m \in M(j) \,:\, \lambda_{j+1,m} \;:=\; & \gamma_{j,m} + 1/2 \textstyle\sum_{k \in A(j,m)} \lambda_{j+1,k} + \\ & 1/8 \textstyle\sum_{k \in B(j,m)} \lambda_{j+1,k} - 1/16 \textstyle\sum_{k \in C(j,m)} \lambda_{j+1,k} \;.\end{aligned}$$

It it immediately clear that `Synthesis` is the inverse of `Analysis`. This is one of the advantages of the lifting scheme. The wavelets constructed here fall in the class of biorthogonal wavelets. If so wanted, one can use the lifting scheme to construct the dual scaling functions and wavelets. For the analysis in this paper, the dual functions are not needed and one can derive the transform by simply making use of the interpolating properties of the scaling functions.

2.6 Calculation of the Integrals

In order to find the coefficients $s_{j,k,m}$, one needs to calculate the integrals $I_{j,k}$ of the scaling functions. This again can be done in a recursive fashion. At the finest level they are approximated with a quadrature formula. Usually, a simple one point quadrature is sufficient. Going from level $j + 1$ to level j is accomplished with the following algorithm:

```
Calculate_Integrals:
```

$$\forall k \in K(j) \;:\; I_{j,k} := I_{j+1,k}$$

$$\forall m \in M(j) \;:\; \begin{cases} \forall k \in A(j,m) \;:\; I_{j,k} \mathrel{+}= 1/2\, I_{j+1,m} \\ \forall k \in B(j,m) \;:\; I_{j,k} \mathrel{+}= 1/8\, I_{j+1,m} \\ \forall k \in C(j,m) \;:\; I_{j,k} \mathrel{-}= 1/16\, I_{j+1,m} \end{cases}$$

This construction ensures that for all levels j the integral of $\sum_{k \in K(j)} \lambda_{j,k}\, \varphi_{j,k}$ is the same.

3 Operators in Wavelet Bases

Consider the computational aspect of applying some linear operator $\mathcal{T}$ to a given function. This is easiest when we can find the eigenfunctions of the operator. For in the basis formed by the eigenfunctions the operator itself becomes diagonal. The cost lies mostly in the transformation back and forth to the eigenfunctions.

The typical example of an operator which is diagonal in space is multiplication with a function. Examples of operators which are diagonal in frequency are convolution (multiplication in the frequency domain), integration and differentiation (multiplication with or division by a monomial), fractional integration and differentiation (multiplication with an algebraic function). The latter relies on the fact that exponentials are eigenfunctions of a differential operator. These operators are used frequently for image processing. One can think of fractional integration as a smoothing operator and of fractional differentiation as an enhancement operator.

Obviously spatially local operators do not need a basis transform. They can be applied directly in the spatial domain. On the other hand, operators which are local in the frequency domain are calculated using the Fourier transform. This picture changes for more general operators. In that case it becomes computationally too expensive to find the eigenfunctions (typically this requires an $O(n^3)$ algorithm).

Wavelets with their support being local in both space and frequency can provide a straightforward and rapidly computable, but at times crude, approximation of more general operators. Localization in space arises from their compact support, while localization in frequency is due to vanishing moments and smoothness. This means that, roughly speaking, wavelets are *approximate eigenfunctions* to operators which exhibit some locality in both space and frequency. A typical example are integral operators whose kernels die off at a particular rate, so-called Calderón-Zygmund operators.

An example from the area of image processing might be an operator which does smoothing or enhancement in different spatial locations. Those operators are typically almost diagonal in the wavelet basis. By only considering the diagonal elements after transformation to the wavelet basis, one can rapidly find a crude approximation to the operator. In many cases, e.g., image processing, this is good enough. Indeed, in those cases the issue is not so much the perfect computation of, for example, the derivative of the given function. Rather, differentiation is pursued since it leads to enhancement. In such cases an approximation of differentiation suffices.

More generally, in numerical analysis applications, one can often use approximate inverses of operators in iterative schemes that converge to the true inverse or one can use them to build inexpensive preconditioners. For historical reasons, it is interesting to point out that people working in abstract mathematics and harmonic analysis already realized some of this in the sixties and at the same time anticipated wavelet-like basis functions. When wavelets came about in the eighties, they provided the perfect computational framework for these ideas. This accounts for part of their immediate success.

Let us go into a little more detail for the case of differentiation. To begin with, consider the classical case where the wavelets are formed by translates and dilates of one "mother" wavelet $\psi(x)$, $\psi_{j,m}(x) = \psi(2^j x - m)$. Now assume that $D\,\psi(x) \approx \psi(x)$. Note that this approximation has to be thought of as very crude. It says nothing more than that a wavelet is a smooth, local "wiggle", and so is its derivative. This now implies that the derivative of a function in the wavelet basis can be approximated as

$$D \sum_{j,m} \gamma_{j,m}\, \psi_{j,m} \approx \sum_{j,m} 2^j\, \gamma_{j,m}\, \psi_{j,m}\,.$$

The approximate derivative in the wavelet basis is thus simply a multiplication with powers of two. Conversely, approximate integration is a multiplication with 2^{-j}. Fractional differentiation and integration would be multiplication with $2^{\pm j\alpha}$. Now assume that we would like to calculate an operator which smoothes in one particular area while performing an enhancement in another area. This would simply result in multiplying the coefficients of the wavelets, whose center of support lies in the former area with a negative power, and the ones whose support lies in the latter area with a positive power. Any number of hybrid schemes can be derived depending on the application.

Spherical wavelets are not formed as translates and dilates of one particular function. In fact they are a typical example of so-called "second generation wavelets" [17]. The idea behind this development is precisely to give up the translation and dilation structure of wavelets, without compromising on the desirable aspects such as localization and fast transforms. This allows construction of wavelets in much more general settings.

For the approximate calculation of operators in this more general context a similar reasoning can be applied. Therefore we consider operators $\mathcal{T}_\beta$ of the form

$$\mathcal{T}_\beta \left(\sum_{j,m} \gamma_{j,m}\, \psi_{j,m} \right) = \sum_{j,m} \beta^{j+1}\, \gamma_{j,m}\, \psi_{j,m}\,.$$

These smooth out if $\beta < 1$ and enhance if $\beta > 1$, and always leave the scaling functions on the coarsest level ($\varphi_{0,k} = \psi_{-1,k}$) untouched. More advanced operators with space and frequency localization can be built by letting β be space dependent. Since the wavelets with $j \geqslant 0$ are constructed to have a vanishing moment, it immediately follows that

$$\int_{S^2} \mathcal{T}_\beta\, f\, d\omega = \int_{S^2} f\, d\omega\,.$$

This insures a desirable property in image processing: the overall brightness of an image is invariant under enhancement and smoothing.

4 Spherical Textures

In this section we consider concrete examples of spherical texture processing and present results achieved with our implementation applying some of the ideas described above. All computations were performed with the lifted Butterfly basis on a spherical icosahedral base mesh subdivided $n = 8$ levels, yielding $10 \cdot 4^8 + 2$ vertices on $20 \cdot 4^8$ triangles. Environment mapping images were ray traced while the texture mapping images were rendered with RenderMan.

4.1 Environment Mapping

A common technique employed in the rendering of highly reflective objects is the use of an environment map [2]. In this technique the radiance reflected across a mirror like surface in the direction of the viewer is approximated by a table lookup in a texture map. The texture map corresponds to a sphere at infinity, giving the incoming radiance as a function of direction. While being only an approximation, which for example, fails with respect to objects close to the reflecting object, impressive effects can be achieved with it. The use of environment maps exhibits the classical tradeoff between computation time and fidelity. Efficiency is bought at the expense of sometimes gross simplification which is deemed acceptable so long as the results are still plausible.

An environment map is typically given as a cubic map, i.e., as six faces of a cube. Since the reflection vector is used as an index into these maps they are effectively point sampled. As with all point sampling techniques care must be taken to avoid aliasing artifacts. Greene [9] suggested the use of MIP maps [19] on each face of the cubic map. The basic idea of a MIP map is to build an image pyramid by recursive averaging over 4 neighboring pixels. Readers familiar with wavelets can think of it as a Haar transform (with only the low pass and not the high pass filters). Filtering each face of the cube separately, however, leads to obvious artifacts at the cube face boundaries. The problems are exacerbated by the fact that proper filtering of a cubic map implies spatially variant filtering, since the desired filter is a function of the distortion between the sphere of directions and its cubic projection. To address this problem, Greene and Heckbert [10] proposed the use of weighted elliptical average filters. They did not, however, use prefiltered environment maps but instead computed each convolution directly. For small filter kernels this is acceptable, but for large filter geometries the performance can degrade quickly and one of the goals of using environment maps, speed, is compromised.

Another application of appropriately filtered environment maps is the approximation of glossy (or mixed diffuse and directional) reflection as pointed out by Greene [9]. Assume a BRDF which is centered around the mirror direction with a symmetric fall-off away from it. Integrating such a BRDF against the environment map is equivalent to the convolution of the map with an appropriately shaped filter. For simple reflection models these convolutions can be precomputed, in effect resulting in an appropriately smoothed environment map. Greene [9, Figure 8] demonstrated this idea for smooth transitions between mirror reflection and diffuse reflection.

The basic task of prefiltering texture maps has received much attention in computer graphics. Examples include MIP mapping [19], filtering by repeated integration [11], summed area tables [3], and decomposition into basis functions [8]. All of these techniques were designed for the processing of *planar* textures.

The main difficulty in applying these techniques to environment maps is that we seek to filter a spherical map, not a planar one. Any mapping from the sphere onto the plane will yield large distortions, e.g., at the poles. Consequently correct filtering implies grossly distorted filter footprints. What is needed instead is a preprocess which filters the original spherical map on a sphere. Typically, this is done for a sequence of increasingly larger filter sizes, not unlike a MIP map. For convenience these filtered maps can then be resampled onto cubic environment maps at appropriate resolution yielding the desired successively smoothed versions of the original map without artifacts introduced by the mapping onto the faces of a cube.

One way to compute these successively smoothed versions is through the use of spherical harmonics. Given an expansion of the texture in terms of spherical harmonics smoothing and enhancement can be applied by proper scaling of the coefficients. However, a number of difficulties are associated with the use of spherical harmonics. Due to their global spatial support they require many high order coefficients to adequately represent sharp features. This also accounts for the high cost of computing expansions of functions in spherical harmonics. Specifically, every sample contributes to every basis function, requiring work which is approximately quadratic in the number of samples. Furthermore application of spatially varying operators would be very difficult.

We have processed an environment map using the ideas of Section 3. In order to visualize the entire environment map we have computed images of the environment reflected in a sphere. This in effect shows the entire map including extreme distortion around the silhouette of the reflective sphere. The original map is shown on the left of the top row in Figure 5. To its right is an enhanced version with $\beta = 1.5$. Note the increased contrast overall and the enhancement of detail in the floor and ceiling. On the right of the top row is an extreme enhancement of $\beta = 2.0$, which brings out much of the noise in the original texture, as one would expect. All images in the top row reflect the processed environment off a perfectly specular sphere. The images in the second row show the simulation of the transition from mirror reflection to diffuse. The model is a mixed model of diffuse plus blurred specular, $(1-\beta)T_0 + \beta T_\beta$, with $\beta = 0.75$, $\beta = 0.5$, $\beta = 0$ (left to right).

4.2 Localized Processing of Textures

To demonstrate the ideas of the local processing of spherical textures we chose a topography/bathymetry data set (ETOPO5) obtained from the NOAA. It gives the height/depth in meters from 0 for the earth. The resolution of the original data set is 5 arc minutes. Due to its large size we first resampled it to 10 arc minutes. A piecewise linear pseudo coloring of the data set is shown on the bottom left of the color plate 5. Note that the height displacements are grossly exaggerated for visualization purposes.

Now consider the following task: Smooth the texture but insure that all coastlines are perfectly reconstructed. This implies that smoothing can only occur away from coastlines. High spatial frequencies can be attenuated already a small distance from a coastline, while low spatial frequencies can only be attenuated at further distances. Coastlines were defined as the zero level set of the original texture $\pm$20m (each triangle at level $n = 8$ has an approximate edge length of 26km).

The basic idea is to split the set of indices $\{(j,m) \mid -1 \leqslant j \leqslant n,\, m \in M(j)\}$ into two disjoint sets: U for the ones that stay **u**ntouched and T for the ones that can be **t**ouched. The operator we use is diagonal in the wavelet basis and looks like

$$\mathcal{T}_\beta \left(\sum_{(j,m)\in T\cup U} \gamma_{j,m}\, \psi_{j,m}\right) = \sum_{(j,m)\in U} \gamma_{j,m}\, \psi_{j,m} + \sum_{(j,m)\in T} \beta^{j+1}\, \gamma_{j,m}\, \psi_{j,m}\,.$$

The question now is how to find the sets T and U. The idea is the following: first flag all finest level vertices which are close to the coastlines. Call this set $\{\mathbf{s}_{n,k} \mid k \in K(n),\, (n,k) \in F\}$. We use the following rule to find F: if for a certain vertex $\mathbf{s}_{n,k}$ the sign of the given data is different at any of its immediate neighbors, or if the absolute value of the data is below 20 meters, then $(n,k) \in F$. The set U now can be found by a recursive algorithm just like `Analysis`, observing that a coarser level basis is in U if any of the bases in its refinement relation are in U. This construction assures that

$$\forall (n,k) \in F \;:\; \mathcal{T}_\beta f(s_{n,k}) = f(s_{n,k})\,,$$

meaning none of the coastlines will have been altered after processing.

The middle image of the third row in Figure 5 shows all triangles at the finest level (green and blue) which intersect the zero contour, i.e., all triangles whose vertices $\mathbf{s}_{n,k}$ have $(n,k) \in F$. Approximately $40,000$ of the vertices (6.1%) fall into this category. Next we determine all wavelets at all levels whose support overlaps the coastline. Their centers, i.e., the $\mathbf{s}_{j,m}$ with $(j,m) \in U$, are shown on the right of the third row in Figure 5. The image demonstrates how the density of basis functions which overlap the coastlines decreases with distance from the coastline. Approximately 142 000 (22%) of all basis functions have support which overlaps some coastline. Note for example that in the Mediterranean most basis functions overlap some coastline. In fact all basis functions at levels $0-3$ overlap coastlines somewhere in their respective supports. Any processing must leave these basis functions unchanged to ensure perfect reconstruction of the coastlines.

The fourth row of Figure 5 shows the unprocessed texture (left) followed by a smoothed version ($\beta = 0.75$). Note how the interior of continents and the bottom of the Atlantic are smoothed out while coastlines are preserved in all their details. This kind of processing would be very difficult to achieve in any non-local scheme. The last image on the fourth row is a rendering of a "soccer-trophy" put (appropriately) in a bar scene. It is a metal sphere that represents the earth; the continents are perfect mirrors and the oceans are mate.

5 Summary and Discussion

We have shown how spherical wavelets can be used to facilitate common texture processing tasks such as smoothing and sharpening for textures which are inherently defined on the sphere. The examples considered the case of a texture defined over a set of directions (environment mapping) and a texture defined over a spherical body (earth topography/bathymetry). Transforming a texture from its nodal representation to the wavelet basis is a linear operation in the resolution of the texture. After this step both smoothing and sharpening operators can be efficiently applied pointwise. The inverse transform then yields the desired result. The local support property of wavelets allows us to apply smoothing or enhancement operators in complex, spatially varying ways.

We have only scratched the surface of the possible texture processing tasks one might perform. It is hoped that the availability of this basic spherical technology will lead to many other applications. Possible future work includes more accurate approximations of BRDFs such as those proposed by Schlick [14] by a sequence of preprocessed environment maps, in effect expressing them as small linear combinations of appropriately shaped filters. Another direction would be segmentation based on wavelet probing [1] as might be useful in satellite based remote sensing.

Acknowledgment

Special thanks to Paul Haeberli of SGI for suggesting environment map processing as an application area of spherical wavelets. Paul Haeberli also provided environment maps and ray tracing software which were used to produce some of the examples described.

The first author was supported by DEPSCoR grant N00014-94-1-1163. The second author was supported by NSF EPSCoR grant OSR-9108 772-004 and ARPA grant AFOSR F49620-93-1-0083. He is also Senior Research Assistant of the National Fund of Scientific Research Belgium (NFWO). Other support came from Pixar Inc. and

DURIP/ONR grant N00014-94-1-0299.

References

1. ANDERSSON, L., HALL, N., JAWERTH, B., AND PETERS, G. Wavelets on closed subsets of the real line. In [16]. pp. 1–61.
2. BLINN, J., AND NEWELL, M. Texture and Reflection in Computer Generated Images. *Communications of the ACM 19*, 10 (October 1976), 542–547.
3. CROW, F. Summed-Area Tables for Texture Mapping. *Computer Graphics (SIGGRAPH '84 Proceedings)*, Vol. 18, No. 3, pp. 207–212, July 1984.
4. DEVORE, R. A., JAWERTH, B., AND LUCIER, B. J. Image compression through wavelet transform coding. *IEEE Trans. Inform. Theory 38*, 2 (1992), 719–746.
5. DONOHO, D. L., AND JOHNSTONE, I. M. Ideal spatial adaptation via wavelet shrinkage. *Biometrika to appear* (1994).
6. DUTTON, G. Locational Properties of Quaternary Triangular Meshes. In *Proceedings of the Fourth International Symposium on Spatial Data Handling*, 901–910, July 1990.
7. DYN, N., LEVIN, D., AND GREGORY, J. A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control. *Transactions on Graphics 9*, 2 (April 1990), 160–169.
8. FOURNIER, A., AND FIUME, E. Constant-Time Filtering with Space-Variant Kernels. *Computer Graphics (SIGGRAPH '88 Proceedings)*, Vol. 22, No. 4, pp. 229–238, August 1988.
9. GREENE, N. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications 6*, 11 (November 1986), 21–29.
10. GREENE, N., AND HECKBERT, P. S. Creating Raster Omnimax Images from Multiple Perspectives Views using the Elliptical Weighted Average Filter. *IEEE Computer Graphics and Applications 6*, 6 (June 1986), 21–27.
11. HECKBERT, P. Filtering by Repeated Integration. *Computer Graphics (SIGGRAPH '86 Proceedings)*, Vol. 20, No. 4, pp. 315–321, August 1986.
12. MALLAT, S., AND ZHONG, S. Characterization of Signals from Multiscale Edges. *IEEE Trans. Patt. Anal. Mach. Intell. 14* (1992), 710–732.
13. MALLAT, S. G. Multifrequency Channel Decompositions of Images and Wavelet Models. *IEEE Trans. Acoust. Speech Signal Process. 37*, 12 (1989), 2091–2110.
14. SCHLICK, C. A customizable reflectance model for everyday rendering. In *Fourth Eurographics Workshop on Rendering*, 73–83, June 1993.
15. SCHRÖDER, P., AND SWELDENS, W. Spherical wavelets: Efficiently representing functions on the sphere. *Computer Graphics, (SIGGRAPH '95 Proceedings)* (1995). To appear, (`ftp://ftp.math.scarolina.edu/pub/imi_95/imi95_1.ps`).
16. SCHUMAKER, L. L., AND WEBB, G., Eds. *Recent Advances in Wavelet Analysis*. Academic Press, New York, 1993.
17. SWELDENS, W. Constructing second generation wavelets using the lifting scheme. Department of Mathematics, University of South Carolina.
18. SWELDENS, W. The lifting scheme: A custom-design construction of biorthogonal wavelets. Tech. Rep. 1994:7, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, 1994. (`ftp://ftp.math.scarolina.edu/pub/imi_94/imi94_7.ps`).
19. WILLIAMS, L. Pyramidal Parametrics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, Vol. 17, No. 3, pp. 1–11, July 1983.