Wavelet Based Texture Resampling

Silviu Borac^{+}
Eugene Fiume^{+*}

$^{+}$Alias|Wavefront and *University of Toronto[1]

Abstract

The integral equation arising from space variant 2-D texture resampling is reformulated through wavelet analysis. We transform the standard convolution integral in texture space into an inner product over sparse representations for both the texture and the warped filter function. This yields an algorithm that operates in constant time in the area of the domain of convolution, and that is sensitive to the frequency content of both the filter and the texture. The reformulation admits further acceleration for space-invariant resampling.

1 Issues

The problem of efficiently processing two-dimensional textures that undergo space variant warps is both important and technically challenging. In most rendering situations, textures are differentially mapped onto surfaces through parameterisation and they are viewed through a perspective projection. Both mappings in principle require texture resampling operations at a cost that can vary significantly from pixel to pixel. The *constant-time space variant* resampling problem is to compute filtered pixel values in image space at a cost that is independent of the *area* of the texture involved in the computation. The importance of efficient resampling algorithms is increasing with the advent of image-based rendering [11].

Most practical approaches use fast but lower quality space invariant algorithms such as those based on MIP mapping [15]. An important design criterion of a general space variant resampler is that it should readily specialise to and accelerate space invariant resampling. Even if the problem of constant-time resampling is solved theoretically, there remains the nagging problem of making the constant as small as possible. Indeed, few "constant-time" resampling techniques are practical, with perhaps the only exception being clamped elliptical weighted average (EWA) filtering on a pyramid [9]. Lansdale demonstrated that with suitable clamping (which itself restricts the degree of space variance), remarkably better

[1]Contact address for either author: Alias|Wavefront, 110 Richmond Street East, Toronto, Canada, M5C 1P1; {silviu|elf}@aw.sgi.com.

visual results than can be achieved than using space-invariant resampling, at only a small additional cost [6]. But truncated gaussians over ellipses are not optimal low-pass filters, and other classes of filters may be desirable. We thus require a technique that is practically efficient in both space and time over filters more general than the gaussians and over filter domains more general than ellipses.

All filtering computations ultimately must perform a convolution operation for each pixel in image space. This involves numerically integrating the product of a filter function with a texture over a potentially large domain. A major source of inefficiency in current schemes is that this integral does not take into account the "information content" of the filter or of the texture: there is in principle no gain if the filter is smooth or if the texture is constant-valued. We would like a representation for the texture and filter that permits a sparse encoding of both so as to minimise the space and time required for computing convolutions.

We shall present a method for texture filtering that is based on a wavelet representation of the texture and of the filter. The approach exploits the orthonormality relations between the wavelet basis functions in order to evaluate the result of the convolution integral of the texture with the filter. The method allows the use of arbitrary filters and the filter shape may vary over the texture domain. We use a wavelet representation to encode the texture, and we approximate the warped filter by a carefully chosen interpolating basis function. This approximation is then itself decomposed into a sparse wavelet representation. A convolution algorithm that operates directly on the sparse representations will then be presented.

The case for a wavelet representation can be made stronger by noting that there is strong evidence that they are excellent for compression. Indeed an easy and effective compression strategy is to use the k wavelet coefficients that are largest in magnitude. This has the advantage of compressing across scales, and it appears to be optimal with respect to RMS error [13]. Such a scheme provides an excellent way for user to specify quite naturally a cost budget for the computation and storage of textures and filters.

The remainder of the paper outlines prior work in the area, a more precise explanation of the problem, a definition of the space-variant resampling problem, a mathematical formulation of our solution, and a suggested set of algorithms.

2 Problem Formulation and Prior Work

The *texture resampling* problem is that of reconstructing a value $I(x,y)$ at any point (x,y) in the image plane through the convolution of a filter function $F_{x,y}$ centred at (x,y) and texture function T. When all $F_{x,y}$ are all identical up to translation by (x,y) to a prototypical filter function F, then the operation over all (x,y) can be written as shift-invariant convolution operator,

$$I(x,y) \;=\; (F * T)(x,y).$$

Depending on whether our domain is discrete or continuous, convolution reduces either to

$$I(x,y) \;=\; \sum_{u,v} T(u,v)F(x-u,y-v),$$

or to

$$I(x,y) \;=\; \int_{u,v} T(u,v)F(x-u,y-v)\,du\,dv.$$

In either case, the homogeneity of convolution admits a dual operation in frequency domain that is just the product of the (discrete or continuous) Fourier transforms of the texture and filter. Except for special situations such as textures derived from spectral noise synthesis, or special classes of band limited filters, frequency domain analysis is usually not feasible.

Most generally (and preferably), we can recognise from the outset that the filters may well vary even within image space. We write the global filtering operation as an integral equation in the form of an inner product:

$$I(x,y) \;=\; \langle T, F_{x,y}\rangle. \tag{1}$$

Texture resampling is complicated by the fact that the viewing operation involves projecting textures from their parametric mapping on surfaces in $\mathbf{R}^3$ back into screen space. In the common case of a perspective projection, the area of the texture projected onto a pixel can vary substantially with position. Because the support of the filter is generally much smaller than the resolution of the texture, it is preferable to project the filter through a pixel, leaving the texture invariant, and performing the resampling operation in texture space. Thus what may have been a space-invariant resampling operation in screen space becomes space-variant in texture space. The mechanics of this process is nicely covered by Heckbert in [9] and is summarised in [8, 5]. The early work on texturing and filtering in computer graphics is surveyed in [10]. Notationally, we can account for this process by making explicit the domain of the space variant inner product in Eq. 1:

$$I(x,y) \;=\; \langle T, F_{u,v}\rangle \mid \Omega_{u,v}, \tag{2}$$

where $\Omega_{u,v}$ is the warped domain of the inner product in texture space that derives from the original domain $\Omega_{x,y}$ in the image plane, and $F_{u,v}$ is the warped filter in texture space corresponding to the original filter $F_{x,y}$ in screen space. The notation $O \mid S$ restricts operation O to domain S.

The EWA filter by Greene and Heckbert presumes that $\Omega_{u,v}$ is elliptical (got through the projection of circular filter supports in screen space), and $F_{x,y}$ is presumed to be gaussian [8]. The technique is not constant-time unless the eccentricity or area of the ellipse is clamped. When combined with MIP mapping [15] and clamping, EWA filtering yields a practical constant-time space variant resampler [6]. Gotsman proposed an alternative constant-time approach, again using gaussians and elliptical extents, that through an SVD analysis steers a filter domain and matches the filter to the local characterisitics of the warp [7].

The most general and most "expensive" of the constant time approaches is *NIL* by Fournier and Fiume [5]. It is general in that it supports arbitrary filters and domains, and it is expensive due to its storage cost and relatively high constant of proportionality at runtime. It is, however, the main point of departure for our technique, so we briefly review its main attributes.

In NIL, an approximate multiresolution representation for arbitrary warped filter surfaces is constructed. For any chosen level of detail, a specific $F_{u,v}$ is approximated as a bivariate piecewise (polynomial) surface

$$F_{u,v} \approx \sum_{\text{patch } P} \sum_{i,j} B_i(u)\, B_j(v)\, b_{ij}, \tag{3}$$

where B_i and B_j are basis functions and the b_{ij} are samples of the filter. Thus for a specific position in screen space, the convolution integral can be written as

$$\begin{aligned} I(x,y) &= \int_{\Omega_{u,v}} F_{u,v} T(u,v)\, du\, dv \\ &= \int_{\Omega_{u,v}} \sum_{\text{patch } P} \sum_{i,j} B_i(u)\, B_j(v)\, b_{ij}\, T(u,v)\, du\, dv \\ &= \sum_{\text{patch } P} \sum_{i,j} b_{ij} \int_{\Omega_{u,v}|P} B_i(u)\, B_j(v)\, T(u,v)\, du\, dv \\ &= \sum_{\text{patch } P} \sum_{i,j} b_{ij}\, C_{ij}. \end{aligned}$$

For a given patch spacing, the set of the C_{ij} defines a quadrature rule for convolution. Once the texture and basis functions are known, this set can be precomputed. Furthermore, a different patch spacing can be used, giving rise to a multiresolution construction for the convolution integral. Fournier and Fiume also present an adaptive quadrature rule derived from recursive patch subdivision.

There are several points of similarity between our proposed technique and NIL. As with NIL, we will approximate a filter surface, this time using an interpolating non-polynomial basis function. However, we decompose both the texture and the filter approximation over an orthogonal wavelet basis. The inner product computation given in Eq. 2 can then accelerated, taking into account both the sparseness of the representation and the orthogonality of the basis.

The NIL construction is storage intensive. The storage required is

$$\left(\sum_{i=0}^{p} \frac{1}{2^{2i}} \right) M^2 RS, \tag{4}$$

where M is the order of the basis functions B_i and B_j in Eq. 3, R is the number of bytes for a C_{ij}, S is the number of texels and p is the number of levels in the multiresolution representation. In contrast, our wavelet representation requires at most RS space, and less if the matrices of the coefficients at different levels are stored in a sparse format. The cost of the convolution operation depends on both the smoothness of the warped prefilter and the smoothness of the texture. NIL does not take advantage of texture regions having low frequency content, but NIL is not alone in this. Using sparse representations for the wavelet decompositions of the texture and the filter actually improves the efficiency of the computation instead of introducing an overhead. The convolution cost is given by the number of pairs of non-zero coefficients of the same type and at the same scale and position.

The sparse matrix format allows to identify this pairs without making comparisons against zero or storing and testing flags.

The method is not limited to the space variant setting. It can be used also as an alternative to prefiltering methods that are less accurate and less costly than NIL, such as trilinear interpolation on a MIP-map pyramid.

3 Choice of Wavelet

There is a bewildering array of wavelet bases to choose from, and the choice must be carefully considered. It is important for the wavelet basis to be orthogonal. The convolution integral is expressed as a sum of dot products of basis functions; choosing an orthogonal basis minimises the number of non-zero terms. It is also desirable to have a small support for the reference wavelet since this economises the computation required to obtain wavelet decompositions.

Some degree of regularity of the basis function is also important. When approximating a smooth function, having smooth wavelets makes the approximation error decay rapidly with the number of basis functions employed. This means that the approximation within some prescribed error bounds will have a smaller number of coefficients. The smoothness and the small support requirements are contradictory. The trade-off we make is to choose the smallest-support orthogonal wavelet that has a continuous derivative. This is the Daubechies wavelet with three vanishing moments, which has two-scale filters of length six. This function is Hölder continuous with an exponent of at least 1.0878 [3] and therefore its derivative is continuous, albeit with a very small Hölder exponent.

An alternative is the spline wavelets introduced by Chui [1]. Since these wavelets can be directly interpolating, they would appear to be well suited to our interpolation problem, obviating the need for the two-step process we described above. However, spline wavelets are only semi-orthogonal, meaning that they are orthogonal across but not within scales. Furthermore, their dual wavelets have non-compact support. Because our two-step process works rapidly and accurately, the small extra overhead is amortised against the economies of orthogonality.

Another alternative is the biorthogonal B-spline wavelets of Cohen, Daubechies and Feauveau [2, 3]. For the special choice of a piecewise linear scaling function we obtain an interpolating function for the filter approximation. However, to be able to perform the convolution by using the orthogonality relations, we need to expand the texture in the basis formed by the dual functions. In the biorthogonal B-spline approach one needs significantly longer filter lengths for the dual functions in order to attain a prescribed amount of regularity: for example C^1 continuity is first reached with a filter length of 13 (see [3], Section 8.3, the spline example with $\tilde{N} = 2$ and $N = 6$).

4 Texture Representation

The first step of the ideal resampling process as formulated by [9] is the reconstruction of the continous texture function $T(\mathbf{u})$ from discrete samples $T(\mathbf{k})$. We achieve this in the first stage of the construction of the wavelet representation by expressing the reconstructed texture in the basis of wavelet scaling functions at the finest scale. We use a standard tensor product 2-D wavelet representation:

$$\begin{array}{rclcrcl}\Phi_{j,\mathbf{k}}(\mathbf{u}) &=& \phi_{j,k_1}(u_1)\phi_{j,k_2}(u_2) && \Psi^v_{j,\mathbf{k}}(\mathbf{u}) &=& \psi_{j,k_1}(u_1)\phi_{j,k_2}(u_2)\\ \Psi^h_{j,\mathbf{k}}(\mathbf{u}) &=& \phi_{j,k_1}(u_1)\psi_{j,k_2}(u_2) && \Psi^d_{j,\mathbf{k}}(\mathbf{u}) &=& \psi_{j,k_1}(u_1)\psi_{j,k_2}(u_2)\end{array}$$

where

$$\phi_{j,k} = 2^{-\frac{j}{2}}\phi\left(2^{-j}x - k\right) \quad \psi_{j,k} = 2^{-\frac{j}{2}}\psi\left(2^{-j}x - k\right) \tag{5}$$

Here, h, v, d are labels for the three detail signals at a given scale, standing for horizontal, vertical and diagonal; further, $\mathbf{k} = (k_1, k_2)$, and $\mathbf{u} = (u_1, u_2)$. Letting $\lambda \in \{h, v, d\}$, the orthonormality relations between the basis functions are:

$$\begin{array}{rcll}\langle \Phi_{j,\mathbf{k}}, \Psi_{j',\mathbf{k}'}\rangle &=& 0, & \text{for } j' \leq j.\\ \left\langle \Psi^{\lambda}_{j,\mathbf{k}}, \Psi^{\lambda'}_{j',\mathbf{k}'}\right\rangle &=& \delta_{jj'}\delta_{kk'}\delta_{\lambda\lambda'}. &\end{array} \tag{6}$$

We use the indexing convention of [3]: a larger scale index corresponds to a coarser resolution level. The largest scale index is denoted by J. The multiresolution ladder is $V_J \subset V_{J-1} \subset \cdots \subset V_2 \subset V_1 \subset V_0$, where

$$V_0 = V_1 \overset{\perp}{\oplus} W_1 = \cdots = V_J \overset{\perp}{\oplus} W_J \overset{\perp}{\oplus} W_{J-1} \overset{\perp}{\oplus} \cdots \overset{\perp}{\oplus} W_1.$$

We build the wavelet representation by first decomposing the reconstructed texture in the basis of wavelet scaling functions at the finest resolution level:

$$V_0 \ni T(\mathbf{u}) = \sum_{\mathbf{k}} T^{0,s}_{\mathbf{k}} \Phi^0_{\mathbf{k}}. \tag{7}$$

Here s indicates a scaling function coefficient and $T_{\mathbf{k}}$ are the wavelet coefficients of the texture. The coefficients $T^{0,s}_{\mathbf{k}}$ can be estimated by using a band-limiting argument (e.g. Prop. 2.1 of [14]): if the functions $T(u)$ and $\phi(u)$ are band-limited to $(-K, K)$, (i.e. the Fourier transforms $\hat{T}(\xi)$ and $\hat{\phi}(\xi)$ are zero for $|\xi| > K$) then

$$\langle T, \phi\rangle \;=\; \frac{1}{2K}\sum_{n=-\infty}^{\infty} T(n/2K)\,\phi(n/2K).$$

This results in the following formula for the coefficients in the decomposition in terms of texture samples and values for the scaling function at integer arguments:

$$T^{0,s}_{\mathbf{k}} \;=\; \langle T, \phi_{0,k}\rangle \;=\; \sum_n T(n)\,\phi_{0,k}(n) \;=\; \sum_n T(n+k)\,\phi(n). \tag{8}$$

The value of K is determined by the texture-sample spacing. The band-limited assumption certainly holds for the reconstructed texture. The wavelet scaling function will have some higher frequency content which decreases with increased regularity. This is a good reason for choosing more regular wavelets.

By applying the standard wavelet decomposition algorithm, we obtain

$$T(\mathbf{u}) = \sum_{\mathbf{k}} T^{J,s}_{\mathbf{k}}\Phi^J_{\mathbf{k}}(\mathbf{u}) \;+\; \sum_{j=1}^{J}\sum_{\lambda}\sum_{\mathbf{k}} T^{j,\lambda}_{\mathbf{k}}\,\Psi^{\lambda}_{j,\mathbf{k}}(\mathbf{u}). \tag{9}$$

At level j every array $T_{\mathbf{k}}^{j,\lambda}$ consists of at most $\frac{S}{2^{2j}}$ elements, but the representation is expected to contain a high proportion of zero-valued elements.

We use a sparse matrix storage format called the *Compressed Row Storage* (CRS) format for the coefficients $T_{\mathbf{k}}^{j,\lambda}$ so as to reduce both storage and the convolution time [4]. In the CRS format, the non-zero elements of the matrix are stored as a vector *vals*. Two other vectors of indices are used to access the matrix elements: *colInd*, which has the same length as *vals* and contains the index of the column, and *rowPtr*, which has one entry per matrix row and indicates the position in the *vals* and *colInd* vectors in which a row starts. We use the additional integers *startRow* and *endRow* to indicate the first matrix row containing non-zero elements. This is important especially for the filter representation since the filter can have its support at an arbitrary position in texture space.

As an example, the matrix

$$\begin{pmatrix} 1 & 2 & 0 & 0 & 3 \\ 4 & 5 & 6 & 0 & 0 \\ 0 & 7 & 8 & 0 & 9 \\ 0 & 0 & 0 & 10 & 0 \\ 11 & 0 & 0 & 0 & 12 \end{pmatrix}$$

has the following CRS representation (indices are zero-based):

startRow	0											
endRow	4											
rowPtr	0	3	6	9	10							
colInd	0	1	4	0	1	2	1	2	4	3	0	4
vals	1	2	3	4	5	6	7	8	9	10	11	12

Note that even when the matrix is not "very" sparse, the added cache coherence afforded by the *vals* array makes it a suitable structure. When we applied a wavelet transform to several common textures, sparse representations of 5%–30% non-zero elements were achieved before noticeable artifacts were introduced.

5 Resampling

We adopt Heckbert's resampling notation [9]: the prefilter in screen space is denoted $h(\mathbf{x})$ (a slight change from Section 2); the texture to screen mapping is $\mathbf{x} = \tau(\mathbf{u})$; the warped prefilter is given by $F(\mathbf{u}) = |\frac{\partial \tau}{\partial \mathbf{u}}| h(\tau(\mathbf{u}))$; the prefiltering operation is $f(\mathbf{x}_0) = \int F\left(\tau^{-1}(\mathbf{x}_o) - \mathbf{u}\right) T(\mathbf{u}) d\mathbf{u}$. The corresponding wavelet representation of F is

$$F(\tau^{-1}(\mathbf{x}_0) - \mathbf{u}) = \sum_{\mathbf{k}} F_{\mathbf{k}}^{J,s} \Phi_{\mathbf{k}}^{J}(\mathbf{u}) \ + \sum_{j=1}^{J} \sum_{\lambda} \sum_{\mathbf{k}} F_{\mathbf{k}}^{j,\lambda} \, \Psi_{j,\mathbf{k}}^{\lambda}(\mathbf{u}). \qquad (10)$$

Using the orthonormality relations (6) the convolution integral may be written as

$$f(\mathbf{x}_0) = \sum_{\mathbf{k}} F_{\mathbf{k}}^{J,s} T_{\mathbf{k}}^{J,s} \ + \sum_{j=1}^{J} \sum_{\lambda} \sum_{\mathbf{k}} F_{\mathbf{k}}^{j,\lambda} \, T_{\mathbf{k}}^{j,\lambda}. \qquad (11)$$

A term in the second sum is non-zero only when both F and T are non-zero. These wavelet coefficients will be non-zero when detail exists at scale 2^j.

```
proc sparseConvolve(filter, texture)
  for row := filter.startRow to filter.endRow do
      fltColPtr := filter.rowPtr[row];
      fltLastColPtr := filter.rowPtr[row + 1] − 1;
      txtColPtr := texture.rowPtr[row];
      txtLastColPtr := texture.rowPtr[row + 1] − 1;
      while fltColPtr ≤ fltLastColPtr ∧ txtColPtr ≤ txtLastColPtr do
            if filter.colInd[fltColPtr] < texture.colInd[txtColPtr]
               fltColPtr := fltColPtr + 1;
            elsif texture.colInd[txtColPtr] < filter.colInd[fltcolPtr]
                  txtColPtr := txtColPtr + 1;
            else
                  Found a match. Add a term to the convolution result.
                  fltColPtr := fltColPtr + 1;
                  txtColPtr := txtColPtr + 1;
```

Figure 1: The convolution iterator.

The coefficients for F are also stored in a sparse format. We use an enhanced Compressed Row Storage format. In the space variant situation, the sparse representation for F must be dynamic because it is generated through a refinement procedure that does insertions of coefficients in an arbitrary order. An iterator is used which identifies non-zero pairs in T and F without testing against zero or using flags. For every scale and h, v, d matrix of coefficients for which we have non-zero coefficients in both the texture and filter representation, we traverse the CRS representations and identify the non-zero products of (11). An outline of the convolution procedure that illustrates how we take advantage of both sparse representations is given in Figure 1.

6 Building the Wavelet Filter Representation

We obtain the wavelet representation of the warped filter in two stages. First an approximation of the filter function is constructed from samples that are taken using an adaptive subdivision algorithm. As in [5], adaptive subdivision affords the approximation of the filter with a bounded number of elements. Then, the wavelet representation of this approximation of the filter is computed. This two-step process seems warranted because the problem of obtaining a representation of a function in a basis of orthonormal wavelets from non-uniform samples is open.

The adaptive subdivision is done with a quadtree algorithm. The nodes of the quadtree are squares with corners at points with dyadic coordinates. For every internal node in the quadtree of an approximation there corresponds a bell shaped interpolating function that takes the value one at the centre of the square and zero on the boundary. The restriction on the corners to having dyadic values for the coordinates is useful because it allows the following important optimization: the wavelet coefficients of the approximating element of a node in the quadtree can be easily obtained by rescaling and translating a precomputed wavelet representation of the unit interpolating function.

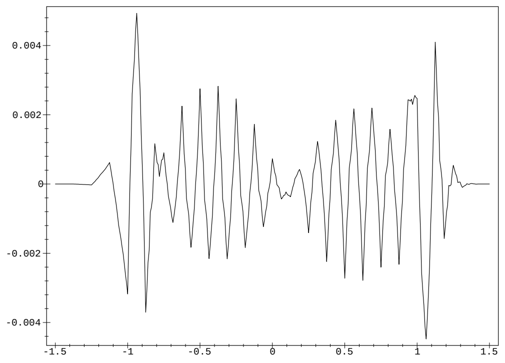

Figure 2: The absolute error for the approximation of ρ by its wavelet decomposition truncated to the first three scale levels.

6.1 Choice of Interpolant

The 2-D interpolant used is the tensor product of a 1-D interpolant: $P(\mathbf{u}) = \rho(u)\rho(v)$ where

$$\rho(u) = \begin{cases} 0 & \text{for} \quad u \in (\infty, -1) \cup [1, \infty] \\ \frac{1}{2}[1 + \cos(\pi x)] & \quad\quad u \in [-1, 1) \end{cases} \tag{12}$$

The reason for preferring the function $\rho(u)$ over the standard tent function is the fact that smoother functions can be approximated with less error by a truncated wavelet decomposition. Low order discontinuities of the function appear with large coefficients over many scales. The function $\rho(u)$ has a continuous first order derivative and second order discontinuities at $u = -1$ and 1. Using (8), the wavelet decomposition of $\rho(u)$ is computed. When limiting it to only three scales we obtain low approximation error with peaks at the points of discontinuity (Figure 2).

The function ρ has also the partition of unity property: $\sum_{k\in\mathbf{Z}} \rho(k) = 1$. This ensures that at least the constant function is represented correctly by a sum of integer translates. This property translates to the 2-D case: $\sum_{\mathbf{k}\in\mathbf{Z^2}} P(\mathbf{k}) = 1$.

6.2 Refinement

The process of building an approximation for the warped filter starts by considering the natural grids of the wavelet decomposition, i.e. those grids with vertices at $2^l\mathbf{k}$ for scale l. We find the scale and position of the smallest grid cell in texture space that covers completely the filter's support. This minimal cell will be the root of the quadtree in the refinement algorithm.

At each refinement level we add a patch of the size of the node in the quadtree and whose shape is given by the 2-D interpolant P. The amplitude of the patch is given by the difference between the value of the warped filter at the centre of the patch and the approximation from the previous refinement levels. The interpolation property of P ensures that patches at the same refinement level do not influence each other. The approximation obtained this way is continuous.

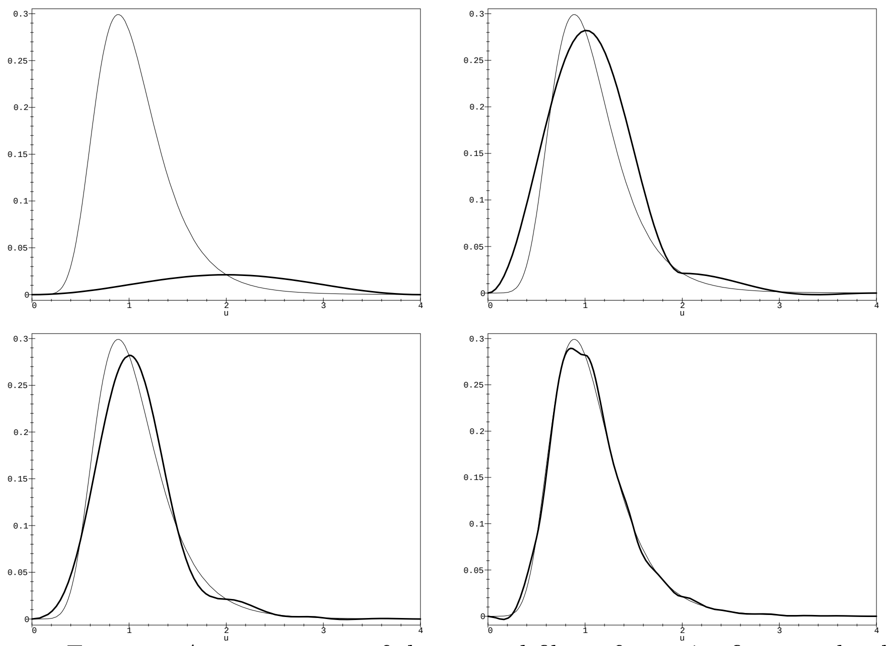

Figure 3: Approximation of the warped filter after 1-4 refinement levels.

The decision to refine a node in the subdivision tree can be based on grid points as in [5], possibly enhanced by using in addition peripheral grid points as in [6].

Figure 3 illustrates the refinement algorithm for a 1-D case gaussian filter warped through a perspective projection. The viewpoint is at unit distance from the line coinciding with texture space. The line of sight makes an angle of 45 degrees with the texture line. The perspective map is $u = x/(\sqrt{2} - x)$, the inverse perspective map is $x = \frac{\sqrt{2}u}{1+u}$ and the jacobian is $|\frac{\partial\tau}{\partial u}| = \frac{\sqrt{2}}{(1+u)^2}$.

Since the refinement quadtree matches at every scale the natural grids of the wavelet representation, we may rescale and translate the precomputed wavelet representation of P by reindexing and then adding it to the representation of the filter. At this point the filter representation is still not suitable for convolution since it may contain scaling function coefficients at various scales and positions. This is a result of the fact that the representation of P contains scaling function coefficients at the coarsest level. A final convolve-downsample [3] sweep from the fine to the coarse levels must be done in order to compute the contribution of these terms to the wavelet representation.

6.3 Space-invariant filtering

The technique can be specialized to the space-invariant filtering case, the main change being that the filter approximation step is no longer necessary. Since we expect the filter approximation to be the most time intensive operation, this would mean a significant increase in performance.

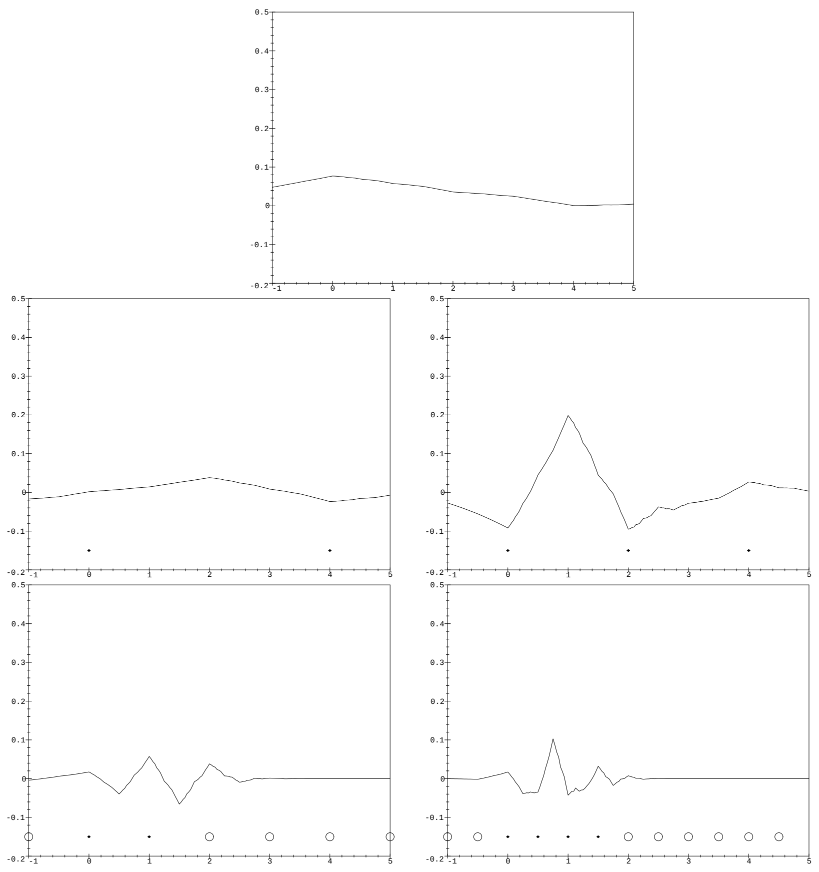
Figure 4: Average and detail components of the approximated filter at four successive scales. The zero entries in the matrix of wavelet coefficients are marked with an empty circle

7 Conclusions and Future Work

This paper has presented a sound architecture for space variant wavelet based texture resampling. The approach further accelerates space-invariant resampling. Our next step is to implement these algorithms in a general-purpose renderer and to study how the process can be further optimised. Special emphasis will be put into improving the algorithm used for building the wavelet representation of the filter. There is a possibility for better integrating the filter approximation procedure with the convolution computation by making use of the auto-correlation functions of compactly supported orthogonal wavelets [12]. These functions are smooth and have the interpolating property; thus they are good candidates for the filter representation. They are also closely related to the orthogonal wavelet basis and therefore the convolution product of the filter represented in the auto-correlation function basis with the texture represented in the orthogonal basis could still be computed efficiently.

Acknowledgements

We happily acknowledge the rendering group at Alias|Wavefront for providing a stimulating environment in which to work and do research. Research funding by ITRC (Ontario) and NSERC (Canada) of the second author is greatly appreciated. We are grateful to the reviewers for their comments and helpful suggestions.

References

[1] C. K. Chui, *An Introduction to Wavelets*, Academic Press 1992.

[2] A. Cohen, I. Daubechies, J.C. Feauveau, “Biorthogonal bases of compactly supported wavelets”, *Comm. Pure Appl. Math.*, 45 (1992), pp. 485–500.

[3] I. Daubechies, *Ten Lectures on Wavelets*, SIAM 1992.

[4] I.S. Duff, A. M. Erisman, J.K. Reid, *Direct methods for sparse matrices*, Oxford University Press, London, 1986.

[5] A. Fournier, E. Fiume, “Constant-Time Filtering with Space-Variant Kernels”, *ACM SIGGRAPH ’88 Conference Proceedings*, also published as *ACM Computer Graphics 22*, 4, (Aug. 1988), pp. 229–238.

[6] R. Lansdale, *Texture-Mapping and Resampling for Computer Graphics*, M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto, October 1989.

[7] C. Gotsman, “Constant-time filtering by singular value decomposition”, *Computer Graphics Forum 13*, 2 (March 1994), pp. 153–163.

[8] N. Greene, and P.S. Heckbert, “Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter”, *IEEE Computer Graphics and Applications 6*, 6 (June 1986), pp. 21–27.

[9] P.S. Heckbert, *Fundamentals of Texture Mapping and Image Warping*, M.Sc. Thesis, Department of Computer Science and Electrical Engineering, University of California, Berkeley, 1989.

[10] P.S. Heckbert, “Survey of texture mapping”, *IEEE Computer Graphics and Applications 6*, 11 (Nov. 1986), pp. 56-67.

[11] L. McMillan and G. Bishop, “Plenoptic Modeling: An Image-Based Rendering System”, *SIGGRAPH 95 Conference Proceedings* (Aug., 1995), pp. 39-46; Annual Conference Series, Addison Wesley.

[12] N. Saito, G. Beylkin, “Multiresolution Representations using the Auto-Correlation Functions of Compactly Supported Wavelets”, *IEEE Transactions on Signal Processing*, special issue on Wavelets and Signal Processing, 1992.

[13] W. Sweldens, “Wavelets, signal compression and image processing”, in *Wavelets and Their Applications in Computer Graphics*, SIGGRAPH ’95 Course Notes, 1995 (http://www.cs.ubc.ca/nest/imager/contributions/bobl/wvlt/download/notes.ps.Z.saveme).

[14] M.V. Wickerhauser, *Lectures on Wavelet Packet Algorithms*, INRIA/Rocquencourt Minicourse Lecture Notes, 1991 (http://wuarchive.wustl.edu/doc/techreports/wustl.edu/math/papers/inria300.ps.Z)

[15] L. Williams, “Pyramidal Parametrics”, *Computer Graphics (SIGGRAPH ’83 Proceedings) 17*, 3 (July 1983), pp. 1-11.