B-Spline Wavelet Paint

Luiz Velho [*]

Ken Perlin [**]

Abstract

This paper describes an unbounded resolution paint system. The image is represented using a bi-orthogonal wavelet basis. The wavelet framework makes it possible to implement the necessary multiscale image operations in an efficient manner and without artifacts. This approach has also many other advantages, including conciseness of the representation.

The architecture of the unbounded resolution paint system is presented and a full 2D implementation is provided.

Keywords: wavelets.

[*]IMPA – Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110
Rio de Janeiro, Brazil, 22460

[**]New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences

1. Introduction

It is desirable and intuitively natural to have a computer paint system that allows for an unlimited amount of detail – as the user magnifies the view, ever more detail is available to be seen and modified. Conversely, as the user demagnifies, large changes can be made quickly over areas that contain a great deal of previously drawn detail.

It turns out that this is not so easy. The most intuitively simple method – using a multiresolution pyramid – does not work. The problem requires a more ingenious technique. In this paper we propose a solution that is based on a B-spline wavelet representation.

1.1 Description of the Problem

An unbounded resolution paint system must support image operations at different levels of detail. The changes to the image at a given level must be propagated to other levels affecting only the appropriate components. The problem consists in defining these multiresolution operations meaningfully and in implementing them efficiently. The key to the solution is to use the right image representation.

1.2 Previous Work

An easier related problem – a bounded multiresolution paint system – was solved by [9]. That system used a multiresolution image pyramid. Operations done by the user at coarser levels were immediately propagated to the finest levels.

The idea of an unbounded resolution paint system using wavelets was first introduced in [7]. Subsequently, an implementation based on Haar wavelets was proposed in [1].

1.3 B-spline Wavelet Paint

In this paper we show why in the unbounded case it is necessary to use a representation that separates the information at different scales, such as the wavelet basis. We describe the principles underlying the unbounded resolution paint system that we have implemented using this approach.

In [7], we discussed the general architecture of an unbounded resolution paint system based on wavelets and gave an example of implementation for one dimensional functions. This paper expands the discussion of a wavelet based paint system and investigates in detail a full implementation in two dimensions.

2. Wavelets

Wavelets are building blocks that can be used as the basis of function spaces, i.e. they allow the description of a function in terms simple elements (or atoms). Such atomic decompositions result in an effective representation of complex data and allow an efficient numerical solution in applications.

The wavelet framework has been used extensively in recent years for image analysis and other scientific applications [3]. Wavelets can be also be used in image synthesis problems such as the multiresolution paint system describe in this paper.

2.1 The Wavelet Representation

Wavelets are families of functions that integrate to zero and are produced by scaling and translating a single function ψ (called mother wavelet)

$$\psi_{a,b}(x) := |a|^{-\frac{1}{2}}\psi\left(\frac{x-b}{a}\right).$$

In order to make the wavelet framework efficient computationally we discretize the scaling parameter a and translation parameter b in the following way:

$$\begin{aligned} a &= 2^{-j}, \\ b &= \frac{k}{2^j}b_0, \quad j,k \in \mathbf{Z}, \end{aligned}$$

Now we have wavelets that are of the form:

$$\psi_{j,k}(x) := 2^{\frac{j}{2}}\psi(2^j x - k), \quad k,j \in \mathbf{Z}$$

where $b_0 = 1$.

A multiscale decomposition of a function f in $L^2(\mathbf{R})$, the space of square integrable real functions, is a linear expansion of f into a countable subset of elements $\psi_{j,k}$.

Wavelets can be used to generate a multiscale decomposition of $L^2(\mathbf{R})$. The idea is to decompose $L^2(\mathbf{R})$ into a direct sum of closed subspaces W_j, spanned by the functions $\psi_{j,k}(x)$.

2.2 Multiresolution Analysis

A multiresolution analysis of $L^2(\mathbf{R})$ consists of a sequence of nested subspaces V_j that are generated by a single function ϕ

$$\cdots V_{-1} \subset V_0 \subset V_1 \cdots$$

whose union is dense in $L^2(\mathbf{R})$ and whose intersection is the null space.

The function ϕ has non-zero integral and is called a *scaling function*. Similarly to the wavelets, ϕ originates by scaling and translation the family

$$\phi_{j,k} = \frac{1}{\sqrt{2^j}}\phi(\frac{x}{2^j} - k).$$

This collection of functions $\{\phi_{j,k} : j, k \in \mathbf{Z}\}$ forms a basis of the approximation spaces V_j. Note that all spaces in the above sequence are scaled versions of the reference space V_0. This means that each space V_j has a “natural” scale 2^j.

In contrast with the wavelet decomposition, in the framework of multiresolution analysis, a function f is represented by its approximations simultaneously at all scales 2^j. Such a structure is composed of several versions of the function f that are computed by projecting f onto the spaces V_j.

2.3 Wavelet Decompositions and Multiresolution Analysis

The complementary subspaces

$$V_j = \cdots \oplus W_{j-2} \oplus W_{j-1}, \quad j \in \mathbf{Z}$$

form a multiresolution analysis of $L^2(\mathbf{R})$ This gives the fundamental connection between multiresolution analysis and wavelet decompositions which, as we will see is the key to fast computation with wavelets.

The scaling function $\phi(x)$ generates the spaces $V_j, j \in \mathbf{Z}$, in the same manner that wavelet $\psi(x)$ generates the spaces $W_j, j \in \mathbf{Z}$. The wavelet functions form a basis of the spaces W_j, while the scaling functions form a basis of the spaces V_j.

The hierarchical data structure containing the wavelet coefficients is called wavelet pyramid. Similarly, the data structure of the scaling coefficients is called multiresolution pyramid.

In this paper we will use the standard convention when describing pyramids [8]. The word ”up” refers to the direction toward greater detail, ”down” to the direction of lesser detail. The bottom-most level of an unbounded image pyramid contains only one pixel. There is no top-most level.

2.4 Wavelet Transform

A family of wavelets may be used to define a functional operator called the wavelet transform. The wavelet transform constitute a powerful framework for the analysis and representation of functions [5].

The discrete wavelet transform of a function f is defined by projecting it onto the elements of the family $\psi_{j,k}$, as in the case of a multiresolution analysis. This gives a representation of f in terms of wavelet family $\psi_{j,k}$.

The goal is to compute the coefficients $d_{j,k} =< f, \psi_{j,k} >$ of the orthogonal projection of a function f onto the wavelet spaces W_j. For this, we resort to decompositions of the approximation spaces V_j. We start with a function $f_J \in V_J$. Since $V_j = V_{j-1} \oplus W_{j-1}$, f_J has a unique decomposition

$$f_J = f_{J-1} + g_{J-1}$$

where $f_{J-1} \in V_{J-1}$ and $g_{J-1} \in W_{J-1}$. By applying this recursively N times we have

$$f_J = f_{J-N} + g_{J-N} + \cdots + g_{J-2} + g_{J-1}$$

for $f_j \in V_j$ and $g_j \in W_j$.

Now, we need an algorithm to implement efficiently the above decomposition and reconstruct the original function from it, (i.e. to perform the direct and inverse wavelet transforms). This is achieved by the fast wavelet transform developed by Mallat [6].

The recursion step in the wavelet transform algorithm exploits the so called *two-scale relations* by which basis elements of V_j and W_j are expressed respectively as linear combinations of basis elements of V_{j+1} and W_{j+1}.

From the two scale relations we have

$$c_{J-1,k} = \sum_n \overline{h_{n-2k}} c_{J,n} = \overline{H} c_{J,n}$$

and

$$d_{J-1,k} = \sum_n \overline{g_{n-2k}} c_{J,n} = \overline{G} c_{J,n}$$

Note that this is essentially a discrete convolution of the coefficient sequence with the filters $\overline{H}$ and $\overline{G}$.

Applying the above formulas recursively on the coarse sequence of approximation coefficients $\{c_{j,k}\}$ we get an algorithm to decompose the finer approximation coefficients into coarser approximation and detail coefficients. Since we are reducing the resolution, the coefficients are decimated by a factor of two (i.e. we keep every other sample). This is also called “downsampling”.

$$\begin{tikzcd}
V_j \arrow[r] \arrow[dr] & V_{j-1} \arrow[r] \arrow[dr] & V_{j-2} \arrow[r] \arrow[dr] & V_{j-3} & \cdots \\
 & W_{j-1} & W_{j-2} & W_{j-3} & \cdots
\end{tikzcd}$$

Going the other way, we have the reconstruction algorithm. Combining the equations $f_{j+1} = \sum_k c_{j,k}\phi_{j,k} + \sum_k d_{j,k}\psi_{j,k}$ and $c_{j+1,n} =< f_{j+1}, \phi_{j+1,n} >$, we get

$$\begin{aligned}
c_{j+1,n} &= \sum_k c_{j,k} < \phi_{j,k}, \phi_{j+1,k} > + \sum_k d_{j,k} < \psi_{j,k}, \phi_{j+1,k} > \\
&= \sum_k (h_{n-2k} c_{j,k} + g_{n-2k} d_{j,k}) \\
&= H c_{j,k} + G d_{j,k}
\end{aligned}$$

The above formula gives an algorithm to reconstruct coefficients of finer approximations recursively from the coarse approximation and detail coefficients. Since we are increasing the resolution, before the convolution it is necessary to interleave the coefficient sequences with zeros. This is also called “upsampling”.

$$\begin{tikzcd}
\cdots & V_{j-3} \arrow[r] & V_{j-2} \arrow[r] & V_{j-1} \arrow[r] & V_j \\
\cdots & W_{j-3} \arrow[ur] & W_{j-2} \arrow[ur] & W_{j-1} \arrow[ur] &
\end{tikzcd}$$

The efficiency of the fast wavelet transform algorithm is due to the fact that, because of the nested structure of the multiresolution analysis, the lower resolution coefficients can be computed directly from the higher resolution coefficients, and vice-versa, without resorting to the $L^2(\mathbf{R})$ inner products.

2.5 B-spline Wavelets

So far we talked about general wavelet functions. When we select a family of wavelets it is necessary to take into account the properties of this particular family which may have impact on the intended application.

Some important properties of the wavelet function are:

Orthogonality: With orthogonal wavelets the fast wavelet transform yields a perfect numerical condition ensuring stable computation of the decomposition and reconstruction algorithms. Orthogonality is very desirable but it imposes a severe restriction on the wavelet functions. An alternative that is more flexible consists in replacing it by a biorthogonality condition.

Support and Decay: If the scaling function and wavelet are compactly supported the fast wavelet transform gives a perfect decomposition – reconstruction scheme. If these functions are not compactly supported, fast decay is desirable.

Regularity: Smooth basis functions are desired in applications where derivatives are involved. Smoothness also corresponds to better frequency localization.

Vanishing Moments: The number of vanishing moments of the wavelet is connected to its smoothness. Vanishing moments are important the rate of convergence of wavelet approximations of smooth functions and in the compression of images.

Symmetry: Symmetry is desirable because makes it easier to deal with boundary conditions and produces better perceptual results for reconstructing images.

Analytic Form: The analytic expression for the scaling function and wavelet is, in general, not available. Nonetheless, the definition of the scaling function and wavelet in analytic form is very useful for computation.

The B-spline wavelets have all the desirable properties listed above: biorthogonality, compact support, smoothness, symmetry, good localization, a simple analytical form in the spatial / frequency domains, and efficient implementation. Additionally, B-spline scaling functions are the natural choice in graphics applications (sometimes even being implemented directly in the graphics hardware).

We have selected a a family of biorthogonal B-spline basis functions with compact support discovered by Daubechies [4]. Biorthogonality implies that we now have two pairs of functions: the wavelet ψ, the scaling function ϕ and their respective duals $\widetilde{\psi}$ and $\widetilde{\phi}$.

In this family, for a preassigned order m of the B-spline ϕ^m there exists infinitely many choices of companion functions $\widetilde{\phi}$, ψ and $\widetilde{\psi}$. We can have different ψ with larger support and different $\widetilde{\phi}$ and $\widetilde{\psi}$ with more regularity.

3. Unbounded Resolution Paint

An unbounded resolution paint system based on multiresolution pyramids (in the spirit of [10]) is straightforward to implement, but it doesn't work. The problem is that painting is a non-commutative operation and it requires lazy evaluation to be practical. Let us examine these two properties in turn.

3.1 Non-commutativity of Paint Operators

To paint a translucent value `(new.color,new.alpha)` over `(pixel.color,pixel.alpha)`, one does an **OVER** operation:

```
new OVER pixel := (
        LERP(new.alpha, pixel.color, new.color),
        LERP(new.alpha, pixel.alpha, 1.0));
```

where `LERP(t,a,b)` is shorthand for the linear interpolation operator $a+t*(b-a)$.

The **OVER** operator is non-linear, and thus non-commutative:

$$(a \textbf{ OVER } b \textbf{ OVER } c) \quad \neq \quad (b \textbf{ OVER } a \textbf{ OVER } c)$$

3.2 Lazy Evaluation

Some sort of multiresolution tree structure is needed to implement an unbounded resolution paint system. The key questions are how often, and in what manner, this tree gets updated.

Suppose you're implementing an unbounded resolution paint system, and the user wants to perform the following sequence of operations:

1. zoom into some fine detail level (level A) and paint a house – covering many pixels at level A.
2. pull back out to a wider view (level B), thereby rendering the house visually tiny.
3. apply a transparent wash over the entire house at level B.
4. view the house again at level A (which will now have its color altered everywhere) and touch up a few details.

The general solution is to store all the information in a tree, and do lazy evaluation - propagating changes from level B to level A only as the user gradually shifts magnification to level A. This guarantees that only a bounded number of pixels need be effected by any single user operation.

3.3 Need for an Orthogonal Decomposition

In order to implement correctly painting operations at unbounded multiple resolutions we need a representation which separates the information at different scales. The wavelet decomposition provides such a representation.

3.3.1 Why not use Multiresolution pyramids?

Given the two properties of non-commutativity and the need for lazy evaluation, multiresolution pyramids are inadequate. The underlying reason is that multiresolution pyramids are highly redundant. Each level contains information that could equivalently reside at other levels. This results in an unfortunate property as information propagates between levels.

Assume that our unbounded paint system uses lazy evaluation - propagating information up the tree only as the user successively magnifies the view.

Consider step 4 in our example scenario of painting a house. As the user zooms back into the house, the information painted at level B will appear to grow successively larger, as information is propagated to successively higher levels of the pyramid. Eventually the level B paint will become larger than the user's visible window.

At some level C intermediate between A and B, information to be propagated will expand past the border of the user's visible window. At this point there will exist two adjoining pixels $P1$ and $P2$ that have information to propagate - $P1$ being just inside the window, and $P2$ being just outside.

Information from level C to level $C + 1$ will be propagated only from $P1$ – not from $P2$ – to the common children of $P1$ and $P2$. If, at some point in the future, the user decides to expand from a view that includes $P2$, information from $P2$ will then be propagated to these common children.

But since painting is a non-commutative operation, the value that ends up at these child pixels will be dependent upon the order of the user's actions. In practice this property produces visible seams in the image at these shared child pixels.

3.3.2 Why wavelets work

Wavelets make it feasible to propagate changes at multiple resolutions in an image without producing the visual artifacts discussed earlier. A image representation based on orthonormal wavelets is non-redundant. In a wavelet pyramid, information propagated from level $L-1$ to level L need not be transmitted to a reconstruction of the image at level L (as is the case for multiresolution pyramids), but only from the wavelet basis elements of level $L-1$ to those at level L. Reconstruction of the image is only done later – linearly and independently at each level.

The image elements at level L are generated by the sum of lower resolution image elements of level $L-1$ with the difference in information between the levels L and $L-1$, represented by the elements of the wavelet basis. Propagation through levels is straightforward. If an image element at resolution $L-1$ is modified, then only the associated wavelet element at level $L-1$ is changed.

In this way, it is possible to employ lazy evaluation as the user moves up and down in the image pyramid. All we need to do is keep at each level of the pyramid a record of changes that have to be transmitted to higher resolutions. Thus it does not matter in what order we propagate information up through the tree. The non-commutativity of the painting operation no longer poses a problem.

4. Paint System Architecture

In this section we describe the architecture of a multiresolution paint system based on B-spline wavelets. This provides correct and efficient manipulation of the image at multiple scales, as well as, a compact representation of the image.

4.1 Overview

The unbounded resolution paint system employs the wavelet representation to encode image data.

The multiresolution painting process consists of a cycle in which the following operations are repeatedly executed:

$$\begin{tikzcd} \textbf{modify image at level } x \arrow[d, bend right] \\ \textbf{move up or down a level} \arrow[u, bend right] \end{tikzcd}$$

The image modification part corresponds to a regular painting operation, affecting only the current resolution level.

Zooming in or out, changes the resolution level and propagates modifications up or down the multiscale structure.

To zoom in, the next higher resolution level $L+1$ is reconstructed from the low resolution component, given by the image elements at level L, and the detail component, given by the wavelet elements at level L.

To zoom out, the current resolution level L is decomposed into a low resolution component and a detail component, both at level $L-1$.

Our implementation based on the above principles contains three distinct parts: an application layer, under that a wavelet hierarchy and a memory management layer. The application layer is what the user sees, the wavelet level maintains the multiresolution image representation and the memory layer implements an unbounded virtual image pyramid.

4.2 Application Layer

The application layer provides an interface for the user, as well as an abstract view of the underlying multilevel structure. This is implemented via a small set of access routines.

In this section we describe the user operations, the access routines, and how the former are mapped into the latter.

4.2.1 User operations

The user is allowed to do only one of three things:

Paint: paint on the current view with some brush

Push: magnify the view by factor of two, scaling about point under the cursor.

Pop: demagnify the view, thereby returning to a previous view.

The main module is

```
Paint_init(canvas_size);
while (not quit) {
  switch (paint_op) {
  case PAINT:
       Drop_paint(draw_path, brush);
  case BRUSH:
       Set_Brush(attributes);
  case ZOOM_IN:
       Paint_change_level(down, cursor_xy);
  case ZOOM_OUT:
       Paint_change_level(up, cursor_xy);
  }
```

Four functions provide the interface to the underlying image hierarchy:
`Paint_init` is called once before the other routines to set up the canvas size.
`Paint_set_value` paints color onto a sample at the current magnification, with opacity alpha.
`Paint_get_value` retrieves the value at one sample at the current magnification.
`Paint_change_level` changes the zoom level by -1 or +1, with the current cursor location as a fixed point.

These functions are part of the wavelet layer and will be described in the next section.

4.2.2 Painting

Painting is done in a single resolution image buffer. This buffer has no knowledge of multiresolution operations. As in most paint systems, the model of the brush is a kernel region of pixels containing a **color** and **alpha**, where **color** is premultiplied by **alpha**. The brush is then applied at the cursor position by:

```
forall xy in region
    image[cursor + xy] := PAINT(image[cursor + xy], region[xy]);
```

where `PAINT(p2, p1)` is defined by:

```
p2.color := p2.color + (1.0 - p1.alpha) * p1.color;
p2.alpha := p2.alpha + (1.0 - p1.alpha) * p1.alpha;
```

There are also auxiliary controls, common to all paint systems, that modify the brush characteristics (opacity, thickness, texture, etc). Thes operations effect only the shape of, and values within, the brush kernel region.

4.2.3 Zooming

Magnification operations can be nested recursively. From the user's point of view, this interface behaves like a LIFO stack. The user "pushes" to magnify the view, and "pops" to return to previously seen, wider angle views that are on the "stack".

This set of operations was chosen for two reasons:

- to keep the conceptual model simple
- to ensure that slow operations (updating the image hierarchy) would not occur while the user was actually painting - updating operations occur only during a **Push** or **Pop**.

For these reasons, we did not implement lateral shifting, in which the user continually slides the view horizontally and/or vertically at a fixed level of magnification. To achieve a lateral shift, the user must successively:

1. **Pop**
2. Move the cursor to the new point of central interest
3. **Push**

The way by which the user is allowed to change resolution levels has important implications in terms of computation and also in the possibility of lazy evaluation of multiresolution compositing.

To implement nested magnification about a fixed point, we maintain a stack offset[] of image offsets. The offset for each magnification level gives the relative (x, y) displacement of the user's view with respect to the underlying image.

Every time the user does a **Push** about fixed point (x, y) on the visible image, we define:

$$\text{offset}[level] := 2 * \text{offset}[level - 1] + (x - x \bmod 2, y - y \bmod 2)$$

Every time the user does a **Pop**, we return to the previous offset$[level - 1]$.

4.3 Multiresolution Layer

The bottom level is the multiresolution data management layer. This corresponds to multiresolution compositing, the image pyramid access functions and the wavelet transform operations.

These operations are only required when the user moves between resolution levels in the pyramid. As we have seen, while the user is painting at a given resolution level the system operates essentially like an ordinary painting program, changing pixel values in an image buffer. This fact makes clear the separation between the top and bottom layers of the system.

In this section we describe how the multilevel image representation is maintained and the mechanism for traversing this structure and performing image composition at multiple levels using lazy evaluation.

4.3.1 Compositing

Multiresolution image compositing is performed by the same routine that changes between levels of the pyramid. This is the fundamental operation of the bottom layer.

`Paint_change_level(delta, center)` Changes the resolution of the paint system and composites the information at the current level with the other levels of the wavelet pyramid using lazy evaluation.

```
Put_image(curr_level)
if (delta > 0) /* magnify */
   propagate mask to next level
   Wavelet_sca_reconstruct(approx_curr, tmp1);
   Wavelet_det_reconstruct(detail_curr, tmp2);
   Composite(tmp1, tmp2, mask_next, approx_next);
   Put_image(next_level);
   clear and store mask at curr_level
} else { /* demagnify */
   Wavelet_decompose(approx_curr, approx_next, detail_next);
   Put_image(next_level);
}
compute new window
Get_image(next_level);
```

4.3.2 Image Pyramid Access

The image pyramid access operations provides the connection between the paint application layer and the multiresolution image data. The pyramid access is implemented by a set of routines that store and retrieve information at one level of the wavelet pyramid.

`Get_Image(level)` This routine stores data from the image buffer in the wavelet pyramid.

`Put_Image(level)` This routine fills the image buffer with data from the wavelet pyramid.

4.3.3 Wavelet Operations

The wavelet operations convert between the the image and the wavelet representations. The image values are associated with scaling function coefficients and the detail values are associated with the wavelet coefficients.

The wavelet operations implement the direct and inverse wavelet transforms as discussed in Section 2.4. The direct wavelet transform decomposes the image into a low-resolution component and a detail component, while the inverse wavelet transform reconstructs the image from these two components.

The two-dimensional wavelet decomposition and reconstruction are separable transformations [2]. Therefore, they can be implemented as two one-dimensional convolutions in the horizontal and vertical directions with the filters H, G, $\widetilde{H}$ and $\widetilde{G}$.

`Wavelet_Decompose(s0, s1, d1)` This routine decomposes the image buffer (`s0`) into a lower resolution (`s1`) and a detail (`s1`) components.

```
1D_wavelet_decompose(horizontal,s0, s1, d1);
1D_wavelet_decompose(vertical,s0, s1, d1);
```

The one-dimensional transform is

```
1D_wavelet_decompose(direction,s0, s1, d1)
{
   for (all blocks in direction) {
      for (all pixels in block)
         s1 <- apply filter H to s0 and downsample
      for (all pixels in block)
         d1 <- apply filter G to s0 and downsample
   }
}
```

`Wavelet_sca_Reconstruct(s1, s0)` This routine reconstructs the image buffer (`s0`) from the lower resolution (`s1`) component.

```
1D_wavelet_sca_reconstruct(horizontal,s1,s0);
1D_wavelet_sca_reconstruct(vertical,s1,s0);
```

The one-dimensional transform is

```
1D_wavelet_sca_reconstruct(direction,s1, s0)
{
   for (all blocks in direction)
      for (all pixels in block)
         s0 <- upsample and apply filter H_til to s1
}
```

`Wavelet_det_Reconstruct(d1, s0)` This routine reconstructs the image buffer (`s0`) from the detail (`s1`) component.

```
1D_wavelet_det_reconstruct(horizontal,s1,s0);
1D_wavelet_det_reconstruct(vertical,s1,s0);
```

The one-dimensional transform is

```
1D_wavelet_det_reconstruct(direction,d1, s0)
{
   for (all blocks in direction)
      for (all pixels in block)
         s0 <- upsample and apply filter G_til to d1
}
```

Note that the reconstructed components `s1` and `s0` are combined in the compositing routine.

4.4 Memory Management

For efficiency of memory access, it is important that locality at each level in the image correspond well with location in memory: two pixels which are very near each other on the image should have a high probability of being near each other in memory. This must be true at all levels of detail.

We implement this as follows. The different wavelet levels are managed independently. Each wavelet level s is managed as a semi-infinite two dimensional "virtual array" $W_s[i,j]$ of wavelet coefficients, where $0 \leq i < \infty$ and $0 \leq j < \infty$.

Memory management of each virtual array is handled by adopting the progressing paging philosophy used by the UNIX file system. We generalize this scheme to higher dimensions.

A virtual array is stored as a tree of pages. Each page contains $res \times res$ pixels, and consists of two halves: The first half contains pointers to descendent pages; the second half contains actual floating point data.

If we let N $= res \times res$, then the pages of the virtual space are organized as follows:

- Page 0 is in the first page.
- Pages 1...N-1 are in the pages pointed to by the first page.
- Pages N...N*N-1 are in the pages pointed to by these pages, etc.

This is illustrated in Figure 1.

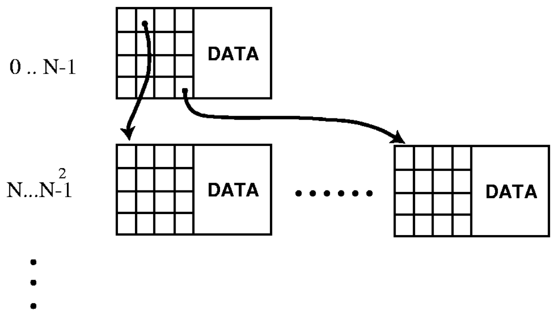

Figure 1:

Note that the first word of the first node is never used in the above scheme. We use this word to store *res*.

We think of all the pages in a virtual array as lying in an infinite grid indexed by I and J, where page I, J contains all pixels such that:

- $res \times I \leq i < (res + 1) \times I$
- $res \times J \leq j < (res + 1) \times J$

We get to page $[I, J]$ in two steps. First, create a radix N number to identify a path down the tree to the page:

```
k := 0
while I > 0 and J > 0 do
   path[k] := mod(I,res) + res * mod(J,res)
   I := I / res
   J := J / res
   k := k + 1
```

Once this path is constructed, we follow it down the tree:

```
page := firstPage
while k > 0 do
   k := k - 1
   if page[path[k]] = NULL
      page[path[k]] := newPage()
   page := page[path[k]]
```

When a rectangle of data is fetched or stored, only those pages containing it are accessed, and created if necessary.

5. Comparison Between B-spline Basis of Different Orders

In this section we compare the effects of using B-spline basis of different orders.

Figures 2-5 show the image of woman's face at various resolutions using B-spline bases of order 1, 2, 3 and 4, respectively (the case of order 1 basis – Haar wavelets – was implemented in [1]). The original image has 300×200 pixels and is associated with level 2 of the multiresolution pyramid. The image is also shown at lower resolutions (levels 0 and 1) and higher resolutions (levels 3 and 4).

5.1 Timings

The computation time of the wavelet decomposition and reconstruction algorithms is dependent of the support size of the filters H, G and $\widetilde{H}$, $\widetilde{G}$ respectively.

The filter supports increase with the order of the B-spline functions.

The relative timings to change between two consecutive levels of the multiresolution pyramid (i.e. performing the direct or inverse wavelet transform) are indicated in the table below (time for Haar wavelet (order 1 B-spline basis) is equal 1):

Basis	*Constant*	*Linear*	*Quadratic*	*Cubic*
Direct Transform	1.0	2.1	2.2	3.1
Inverse Transform	10.6	19.1	20.2	27.4

5.2 Analysis

The above examples show that the use of higher order B-spline scaling functions and wavelets produces better results. Magnification benefits from higher order low pass filters producing an effect that is equivalent to anti-aliasing. Demagnification benefits from higher order high pass filters producing an effect similar to sharpening.

Higher order B-spline functions also imply in more computation.

We found that a compromise solution that offers a good quality / speed trade-off is obtained with the functions of order 3 (quadratic B-spline).

6. Examples

In this section we show an example of using the multiresolution paint program. In the example we employ a quadratic B-spline. **Rectangle over eye**: In this example we paint a translucent rectangle over the eye of a woman. This illustrates the effect of multiresolution compositing with transparency. Figure 6 shows the face with the square painted over the eye at low resolution and the eye at higher resolution. Note that all details of the eye are preserved.

7. Conclusions

We described an unbounded resolution paint system using based on wavelets. The image is represented using a bi-orthogonal wavelet basis. This framework makes it possible to implement the necessary multiscale image operations in an efficient manner and without artifacts.

References

[1] D. Berman, J. Bartell, and D. Salesin. Multiresolution painting and compositing. Technical Report 94-01-09, University of Washington, Seattle, Washington, 1994.

[2] E. Catmull and A. R. Smith. 3-d transformations of images in scanline order. *Computer Graphics (SIGGRAPH '80 Proceedings)*, 14(3):279–285, July 1980.

[3] C. K. Chui, editor. *Wavelets: A Tutorial in Theory and Applications.* Academic Press, 1992.

[4] I. Daubechies. *Ten Lectures on Wavelets.* Number 61 in CBMS-NSF Series in Applied Mathematics. SIAM Publications, Philadelphia, 1992.

[5] S. G. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Trans. on Acoust. Signal Speech Process.*, 37(12):2091–2110, 1989.

[6] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 11(7):674–693, 1989.

[7] K. Perlin and L. Velho. A wavelet representation for unbounded resolution painting. Technical report, New York University, New York, 1992.

[8] S. L. Tanimoto. Image data structures. In S. L. Tanimoto and Klinger, editors, *Structured Computer Vision.* Academic Press, New York, NY, 1980.

[9] Lance Williams. Mip paint system. NYIT, 1982. Personal Communication.

[10] Lance Williams. Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):1–11, July 1983.

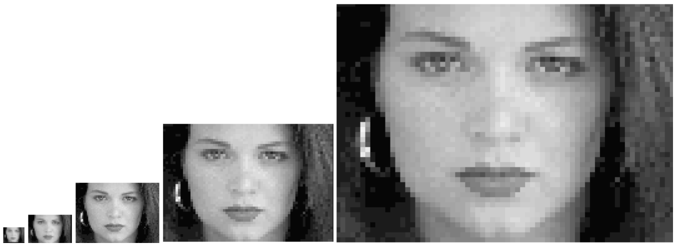

Figure 2:

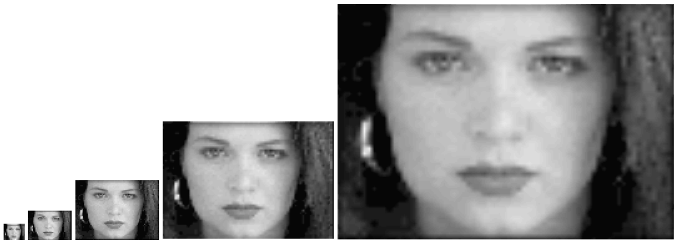

Figure 3:

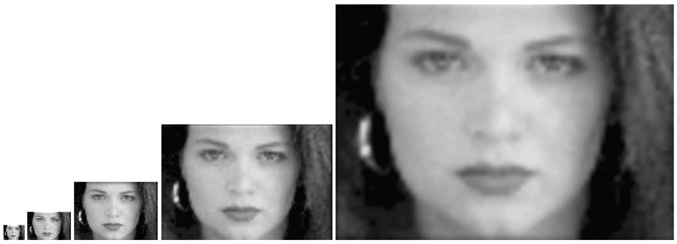

Figure 4:

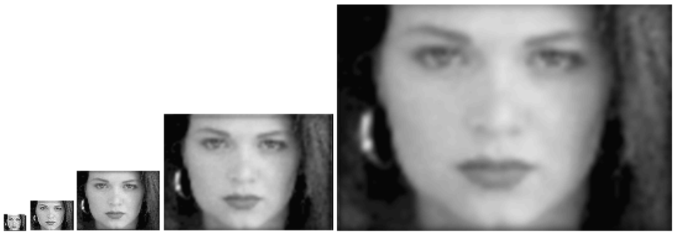

Figure 5:

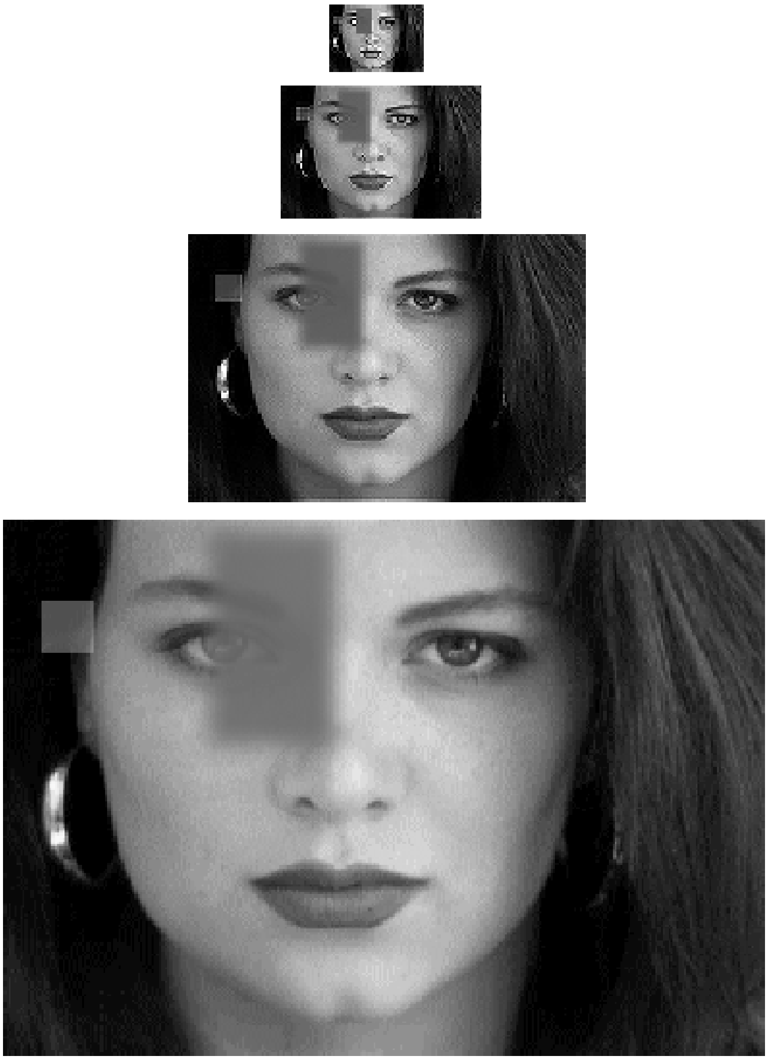

Figure 6: