A Difference Engine for Images with Applications to Wavelet Decompression

E.H. Blake

A.A.M. Kuijk

Centre for Mathematics and Computer Science (CWI),
Department of Interactive Systems, Kruislaan 413,
1098 SJ Amsterdam, The Netherlands.
Email: edwin@cwi.nl

15th July, 1992

Abstract

The low level components of a new raster graphics architecture have proven to have novel uses in image reconstruction. The display hardware can be regarded as a very fast (11ns per operation) difference engine that works in two-dimensions. The speed is partly achieved by the use of custom VLSI components for the most primitive operations and this permits the video rate reconstruction of images and other signals compressed by encoding them on various polynomial bases. The paper describes the architecture and its operating parameters.

The results are presented of a feasibility study into the use of the system as an image decompression system. The study also shows that the system can be applied to the decompression of spline wavelet encoded images.

1 Introduction

A radical reappraisal of the three-dimensional (3-D) interactive raster graphics pipeline has resulted in an experimental architecture for a graphics workstation which is currently being evaluated at the CWI. Some of the novel uses of parts of the hardware were not foreseen when the research project was initiated. In this paper we shall explore these unanticipated spin-offs from the project.

The principal features of the design for the new raster graphics architecture are:

1. Emphasis on real-time interactive shaded 3-D graphics.
2. Object space methods rather than image space methods are used where possible.
3. Avoids the use of a frame buffer.
4. Uses custom VLSI only at the lowest, most primitive, levels where commercial products are unlikely to suffice in the near term.

It was these design decisions that lead to a number of interesting consequences that have made parts of the architecture eminently suited to a a far wider range of problems in computer graphics and image processing[1].

For example, the custom VLSI development that was a major part of the project produced what is essentially a very *fast difference engine* (to borrow a term from the 19th century history of computation). This engine can compute forward differences in parallel over the whole width of a typical image, taking about 11ns per operation (90 Mhz clock) independently of the length of the forward difference spans.

In the next section we present a very brief overview of the architecture in terms of its original design for producing real-time interactive raster graphics. In Section 3 we show how the same low-level architecture has applications in the reconstruction of compressed signals, and we present the results of an initial feasibility study. In Section 4 we introduce wavelets as a new generalized and promising approach to studying compression and decompression schemes for sounds and images. In conclusion (Section 5) we summarize our results and point to the some very promising areas for further research into applications of the hardware under discussion.

2 Overview of the Raster Graphics Architecture

A number of different feedback levels in the image synthesis pipeline can be identified if one takes a new look at the basics of high quality three-dimensional raster graphics [27]. From the highest to the lowest level these include:

- Modelling
- Coordinate Transformations
- Viewing Transformation
- Hidden Surface Removal
- Area Primitive Processing

[1] Without going into a contemplation of the meta-levels of the design process it is interesting to observe that this generality of application resulted from bottom-up design. The initial top-down design produced an architecture for raster graphics (only). The bottom-up design that followed concentrated on extracting the lowest common denominator of primitive operations for synthesizing pixels — a language for manipulating related pixels. This vocabulary can now be used for expressing other facts about images.

Visual effects can only be achieved by interaction with the data at each of these levels. A user interacts exclusively with the visible parts of a 3-D model, but not with pixels since they are too primitive a kind of object to be of interest to a user. We provided direct access to graphics objects to support pointing and identification. These actions are considered fundamental because they underlie every change a user makes in a picture.

Changing pictures form the key to the architecture. Actual pixels are not needed for user interaction. If we take this observation seriously and ruthlessly pare away other elements we get a radical prescription for a graphics architecture. One where the visible surfaces of objects are explicitly identified, and without any mandate for a frame buffer. We believe that our research shows that such a machine, which harkens back to the calligraphic roots of graphics displays, can be built [1].

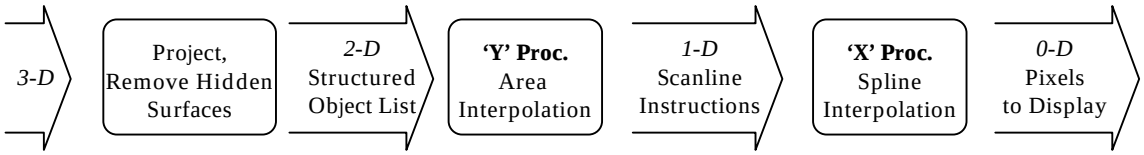

Figure 1: Functional Elements of Display Architecture.
The *Y* and *X* processors together form the display controller. The arrows show how 3-D data are converted to a stream of pixels. Data structures are indicated by arrows and of these only the 3-D Models and the 2-D Structured Object List are stored. All other data are generated on the fly. The Structured Object List is updated at the animation or interaction rate, 12–24 times per second, and it is read at the video frame rate, 50–60 cycles. The Scanline Instructions are produced at the video line rate and clocked into the *x* processor at the video pixel generation rate.

In the past four years the design has been completed [19], the critical components have been made [18] and functions simulated in detail [17]. The rest of this section presents an outline of the architecture. The functional elements of the display architecture are shown in Figure 1. The major new components follow from the *bottom up* (right to left in Figure 1):

1. pixel generator (or *X* processor — Section 2.1).
2. area processor (or *Y* processor — Section 2.2).
3. hidden surface removal, projection and other higher level functions, that are not discussed here at all.

Items 1 and 2 together form the display controller. In our architecture the frame buffer is replaced by a structured list of objects which can be *pointed at* (at the video frame rate of 50–60 cycles). To this end a powerful VLSI-based hardware block containing a systolic array of processors produces a full colour pixel stream at video refresh rate. This block is fed with instructions produced by general purpose microprocessors from the structured object list. The processors are capable of producing Phong shaded 3-D objects or 2-D textures at this rate. For full colour images one systolic array is needed for each of the primary colours (red, green and blue)[2].

The structured object list contains objects that describe small 2-D surface primitives — things like triangle meshes, trapezoids or splines outlining characters in various typefaces. The objects have all the information necessary to render them on the display. The size of the object list does depend on the complexity of the image. The size of the list is about the same as a conventional frame buffer, with the advantage that the object list is resolution independent so that the list will not become larger for higher resolutions. The list is also a much more organized data structure and can therefore support more sophisticated operations than a simple frame buffer.

[2]It is equally possible for the processors to implement another colour model internally and use converters on the output side to produce the required values. Postprocessing of this kind can also detect underflow and overflow and substitute default output values.

At higher levels of the architecture the objects become more complex (but also fewer — less fragmented) to maintain information about light sources, textures and viewing (hidden surface removal or overlay priorities etc.). Here representations for incremental changes typical for real-time interaction are favoured. These requirements appear to be satisfiable in the short term by powerful but "off-the-shelf" parallel hardware. At the lowest level custom components are needed: these have already been built.

2.1 The Difference Engine

The processor referred to above as the pixel processor or *x* processor (because it deals with the scanlines of an image) is actually a difference engine. That is, it can do forward difference calculations at high speed. Any order of forward difference can be done, the limiting factor is the accumulation of errors during addition and (in real-time applications) the extra cycle time taken by each higher order. The architecture is tailored to second order forward differences.

The processor is a systolic array with a dedicated processor for each pixel in a scanline[3]. Additions (and input) are done to a precision of 36 bits and the top 12 bits are output. So we basically have data values of 12 bits, say 10 bits magnitude and 1 sign bit and 1 bit to detect common overflow situations, and 24 precision bits. For quadratic interpolation (second order forward differences) spans of 2^{12} bits or 4096 pixels wide can be interpolated before the error becomes noticeable. This is also the maximum addressable pixel and span width allowed for input. If we use just 9 data bits of the output (i.e., 8 bit values + sign bit) then cubic interpolation for spans of 512 pixels can be done accurately. Naturally longer spans can always be done by splitting them into shorter ones which can be done correctly.

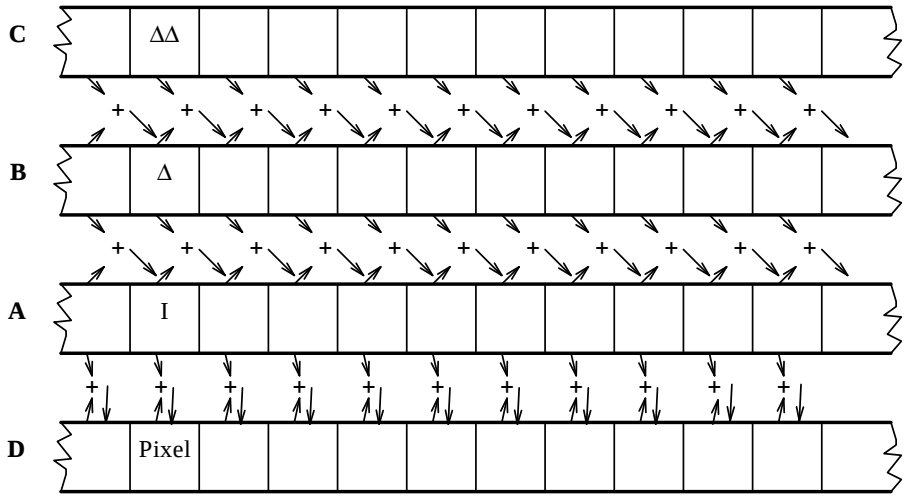

Figure 2: Interpolation on the Difference Engine
The difference engine is implemented as a systolic array of processors: one for each output pixel. For quadratic interpolation the second differences remain constant and are propagated to all processors within an active span of pixels (Register C). The first differences (Register B) are changed by the second differences at each step and the results added to the intensity values (Register A). The results of the interpolation step are added to an accumulator (Register D) so that multiple interpolation spans may overlap to produce the final pixel value. For higher order differences the same registers are reused.

The bias towards quadratic interpolation is also seen in the fact that there are 3 internal registers with which the value I ("intensity"), first difference δI and second difference $\delta\delta I$ can be set at every pixel location. The operation of the difference engine is illustrated in Figure 2. The

[3]Actually one can have fewer processors provided the number of output pixels are a multiple of the number of processors. The processors are then reused for the remaining parts of a scanline, but the normal situation is to have one processor per pixel in a scanline.

operation	description	cycles
acc_mode	Accumulate mode: if enabled negative intensities are not added to accumulator	1
dis(x,dx)	disable accumulation of intensities from pixel 'x' for 'dx' pixels (cleared after next 'eval*' command)	1
eval0(x,dx,i)	Set pixel (i.e., accumulator) from 'x' for span of 'dx' directly, disable further additions until next refresh.	1
eval1(x,dx,i)	add i to accumulator for span from 'x' for 'dx' pixels	3[a]
eval2(x,dx,i,di)	First order forward difference, starting at pixel 'x' with value 'i' and increment 'di', for 'dx' pixels	4[a]
eval3(x,dx,i,di,ddi)	Second order forward difference — like 'eval2' except now 'di' is also changed by 'ddi' at each step	5[a]
eval4(x,dx,i,di,ddi,dddi)	Third order forward difference, like 'eval3' *mutatis mutandis*	6[a]
eval n	Higher, $n-1$, order forward differences	$n+2$[a]
nop	No operation	1
refresh	Output accumulator value and clear everything	2
setddi(x,dx)	Set (i.e, override) second difference[b] at points 'x', for a span of 'dx' pixels in the middle of the next 'eval' command	2
setdi(x,dx,i)	Like 'setddi' only it affects the lower forward difference	2
seti(x,dx,i)	Like 'setdi' except that this creates a span of intensities	2
setpddi(x,dx)	Set (i.e, override) second difference[b] at points 'x', 'x+dx', 'x+2dx', ... in the middle of the next 'eval' command	2
setpdi(x,dx,i)	Like 'setddi' only it affects the lower forward difference	2
setpi(x,dx,i)	Like 'setdi' except that this creates a pattern of intensities	2

Table 1: *X* Processor Instructions and Their Costs in Cycles
Note: The costs mentioned above are incurred whether a span is 1 pixel long or covers the whole width of the scanline.

[a] The cost of this operation can be reduced by 1 cycle in future versions
[b] If there are higher order differences then this sets the highest order difference

interpolated intensities are calculated in register A and the contents of A is added to an output accumulator which can contain the results of a number of previous interpolation spans.

The contents of the accumulator is read out (for display) on receipt of a "refresh" instruction. Other instructions are listed in Table 1.

Higher order differences are done by reusing the registers. In fact the registers are not needed to perform forward differences but only so that differences may be set ahead of time by means of the 'set' instructions (see Table 1). It is a way of changing just certain (usually the highest) differences within an interpolation that is continuous in the lower orders.

To conclude this *x* processor which was developed as a very specialized pixel generator is really very general; it is a *Difference Engine* with:

- Any order forward differences.
- 36-bit numerical accuracy and an 11ns cycle time.
- Any spline interpolation, with constant cost independent of span length.

2.2 'Y' Processor

The information to drive the area or *y* processor is contained in the Structured Object List (see Figure 1). It contains the visible areas of the displayed objects and their colouring information. This processor has the basic task of producing the instructions for the arrays of pixel processors.

The shading processors operate at the frame refresh rate and go through a complete cycle once every video frame. Their input data are produced at the interaction or animation rate (between 12 and 24 cycles per second). The output goes to the pixel processors operating at the line refresh rate.

The task of these processors can also be described as having to change 2-D display information into 1-D scanline information. The third dimension has been dealt with at an earlier stage by projection and hidden surface removal. The geometry can be specified solely in terms of (2-D) display coordinates but (3-D) world coordinates are still needed for the vectors which underlie the shading calculations.

The edges of the surface primitives (triangles or trapezoids) are simple enough to find. Shading (in particular Phong shading) and anti-aliasing are more of a challenge. We have to recast the Phong shading model, but without lapsing into expensive Gouraud shaders or being unable to deal with all practical situations. Phong shading itself is nothing more (or less) than a very good practical approximation — it is not an end in itself. In [20] we introduced a method for quadratic Phong shading via angular interpolation. It has the lowest per pixel cost in time and storage of any method we are aware of.

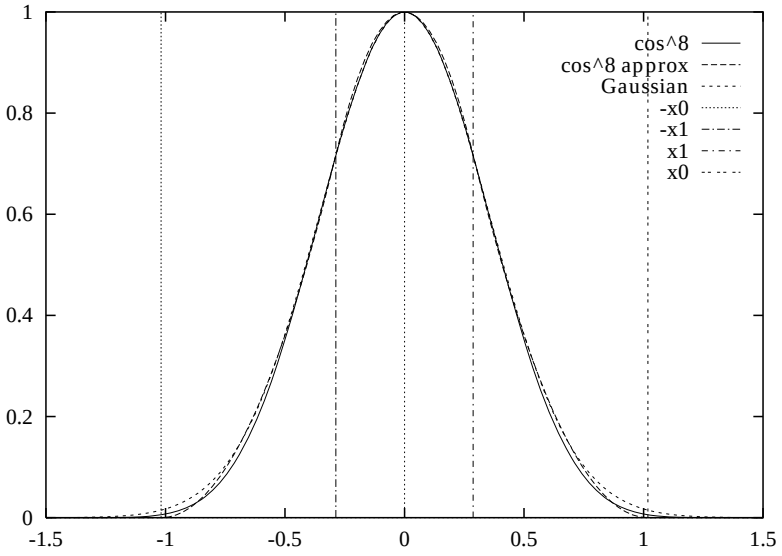

Figure 3: Quadratic Approximation of a Cosine to a Power Used in for Phong Shading
The diagram shows the curve of $\cos^n$ and the quadratic spline approximation of the same function, here $n = 8$ but the approximation is valid for wide range of n. The knot points of the splines are at $-x_0$, $-x_1$, x_1 and x_0. A Gaussian curve (i.e., a function of the form e^{-x^2}) is shown which has the same value at the points $-x_1$, 0 and $+x_1$ as the other curves.

Our approach to quadratic Phong shading depends on two major results:

1. A parameterized piecewise quadratic expression for $\cos^n \theta$ — the cosine of an angle θ, $-\pi/2 < \theta < \pi/2$, raised to a power $n, 1 \leq n \leq 125$. The shape of the curve is very similar to a Gaussian (Figure 3).

2. A linear expression in terms of the pixel position, x, on a scanline for the angle θ between the interpolated normal vector of the surface and the light or highlight vector.

Anti-aliasing in the form of exact area integration is applied. Object space hidden surface removal preserves the necessary information on pixel coverage. Where pixel coverage gradually

increases along a scanline this is treated as a linear modulation of the quadratic shading function. This results in a cubic expression which is passed on to the pixel generator.

The instructions generated by the area processor are thus expressions that describe various splines that are to be interpolated. An area primitive has been coded in terms of a spline basis.

Two-Dimensional Quadratic Interpolation

It is possible to encode the shading functions of whole trapezoids as 2-D quadratic functions. The following information is required (although the last few terms may not be necessary):

start value	I
y increment of start value	$\frac{\delta I}{\delta y}$
2nd differences of the y-increments	$\frac{\delta^2 I}{\delta y^2}$
1st x difference	$\frac{\delta I}{\delta x}$
y increment of x difference	$\frac{\delta^2 I}{\delta y \delta x}$
2nd x difference	$\frac{\delta^2 I}{\delta x^2}$
y-increments of the 2nd x differences	$\frac{\delta^3 I}{\delta y \delta x^2}$
2nd differences of y-increments of the x differences	$\frac{\delta^3 I}{\delta y^2 \delta x}$

Needless to say the higher order interpolation puts severe demands on the numerical precision of the interpolations.

3 Reconstruction of Compressed Signals on a Difference Engine.

Clearly the *x* processor can interpolate any spline (polynomial) curve. Thus any signal that is expressed in terms of a *spline basis* can be reconstructed. Not only that, the architecture with its accumulator allows one to sum over incrementally generated output so that the splines can be summed over different scales to produce the final image to any required accuracy.

3.1 Simple Compression

Obviously run length encoding is handled by a single instruction — 'eval0', see Table 1. One could also extend the concept to using linearly increasing runs, although the benefit is doubtful.

The intended benefit of compression is to save both space and time, space is saved since images become smaller and the transmission times also decrease. The cost is of course having to perform the compression step first and then the decompression. In a number of situations the most time critical step is decompression, where images have to be viewed in (near) real-time. The difference engine can greatly assist with this as it can interpret the actual image coding instructions directly[4].

3.2 Expressing an Image in a Multi-Resolution Polynomial Basis

A simple experiment has been run to demonstrate how the difference engine can be used reconstruct an image expressed in a multi-resolution polynomial basis. The interest of the demonstration is to show how reconstruction would work in practice — not (yet) to uncover a new

[4]Image compression steps are frequently followed by more traditional and well understood data compression algorithms, we do not discuss those here.

compression technique. The image encoding method will be described below. The results of the encoding was a stream of *x* processor instructions that could be truncated at any desired resolution and consisted of full size descriptions of the image at each resolution level. A number of these are shown in Figure 4. These instructions were interpreted by a fast systolic array simulator to recreate images which were more or less blurred as desired (see Figures 6–8).

Figure 4: Linear, First, Second and Seventh Quadratic Terms
Individual component images at the various levels of encoding the test image. The image size was 256 by 256 pixels. Clockwise from top left we have: (a) linear term, (b) first parabola, width 256 pixels, (c), second parabola, and (c) second last parabola of width 4 pixels. The cumulative effect of these summing terms is shown in Figures 6–8.

A Simple Encoding Algorithm

To illustrate the usefulness of the architecture introduced in Section 2 a multiresolution encoding of the image was needed. A simple algorithm was developed for test purposes which we believe illustrate the essential ideas while having a very low computational cost. It works only in 1-D along scanlines and requires no convolutions[5]. The image can have any size within the address range of the *x* processor.

Since compression was not the objective a very simple basis function was chosen: a parabola of the form $1-x^2$ defined on the domain $-1 \leq x < 1$. This length is then scaled to spans of width 2^n, where $2 \leq 2^n < 2\times$ (width of image-1), $n \in \mathcal{Z}$, and translated in steps of 2^n where necessary to tile the whole scanline. There are various small optimizations to possible to handle symmetries and very short spans. The operation of the algorithm on a single scanline is illustrated in Figure 5, the steps are as follows:

[5]The astonishing thing is that this algorithm actually produces a useful image decomposition.

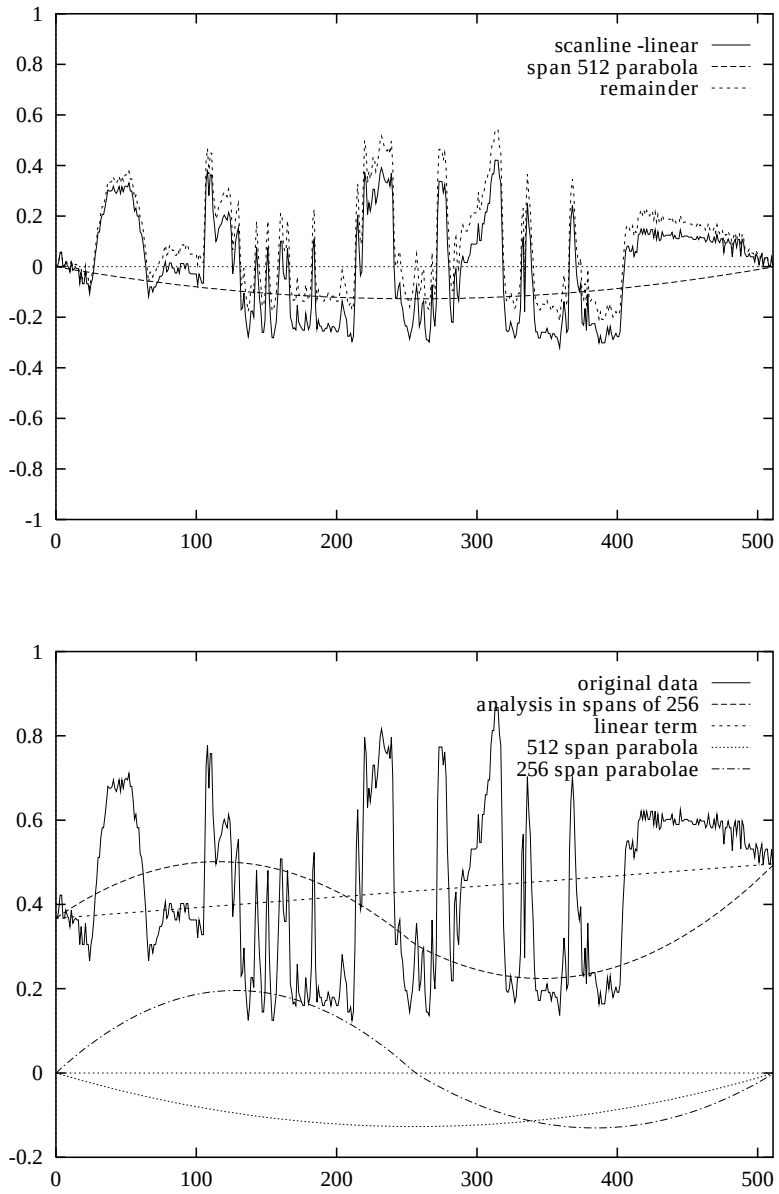

Figure 5: Encoding A Single Scanline

The diagrams illustrate the situation in encoding a single scanline that contains the eyes from the Figure 6. The top diagram is the situation where the first parabola is “fitted”. This has a span of width 512, the parabola of width 1024 would have been needed if the image was just one pixel wider. The lower diagram shows the results, and the basis functions, of “fitting” spans 512 and 256 pixels wide.

- The "compression" algorithm first finds the linear trend in a scanline and produces the instructions which would interpolate that linear term (Figure 4).
- The linear span is subtracted from the input image. This produces the difference image which is illustrated on the top left in Figure 6.
- Subsequently the first quadratic span whose length is a power of two and whose centre falls within the image width is chosen, i.e., span length = 2^n where 2^{n-1} describes a pixel inside the image, see Figure 5. The value of this pixel *alone* determines the height of the parabola (no convolutions or other complications).
- As with the linear term, the scaled values of the parabola are subtracted to produce a new difference image.
- The next span length is half the previous one and the procedure is repeated recursively. Note that the one pixel whose value was chosen is never changed by higher resolution (smaller) parabolae.

Results

The results of applying the procedure and then reconstructing the images on the *x* processor simulator are illustrated in Figures 6–8. It can be seen that it is possible to decode a low resolution image directly and that the natural way of reconstructing the image is to build up the output resolution level by resolution level. In a real-time application one could therefore have graceful degradation of performance since the display controller could truncate the image being processed after a certain number of instructions and one would still have a useful image. In some pyramid systems this is more of a problem since the higher levels of the pyramid (the lower resolutions) actually are smaller images and need to be expanded for display purposes.

The compression procedure outlined above could be turned into a true compression of the image by adopting a pyramid compression scheme as outlined in [3] but coding it explicitly in terms of the basis functions..

One feature of note in our simple coding scheme is that the actual edge information is retained far too long. This visually important information should really be sent early on with the low resolution image. Other more sophisticated compression schemes also suffer from this problem. Thus an important area of investigation is to find compression algorithms that send "important" information first, not merely low resolution information. One could also conjecture that such algorithms will also be more efficient as lossy encoders of images destined for viewing by people (see also Sections 4.3 and 5).

4 Wavelets

Wavelets are a "new" decomposition tool for analysing signals at multiple scales. Multi-resolution analysis has been around for a long time but recently there has been significant advances both in unifying old ideas in a single theory and in deriving new results. Indeed a recent introduction to the topic has referred to a [10]:

> ... this comparative frenzy of research activity ... which has provided not only a wealth of new mathematical results, but also a common language and rallying call for researchers in a remarkably wide variety of fields: mathematicians working in harmonic analysis ...; mathematical physicists ...; digital signal processors because of connections with multirate filtering, quadrature mirror filters, and subband coding; image processors because of applications in pyramidal image representation and compression; researchers in computer vision who have used "scale-space" methods for some time; researchers in stochastic processes interested in self-similar processes, 1/f

Figure 6: Image on Parabola Basis, Steps 0–2.
The images on the left are the remainders after the encodings on the right have been subtracted. On the left images have been scaled so that normalized pixels with the value of -1 are black and +1 are white. On the right 0 is shown as black and 1 as white, while negative values, which can exist in intermediate images, are suppressed. The right hand images were executed on a simulation of the difference engine described in Section 2.1.

noise, and fractals; speech processors interested in efficient representation, event extraction, and mimicking the human auditory system. And the list goes on.

Wavelets or "the principles of multiresolution analysis" are still the subject of intense research. It is however clear that it shows great promise for image compression, see Section 4.3. Indeed this kind of analysis (as can be seen in the above quote) has quite a long pedigree in image processing dating back to the early eighties [3, 4, 32]. What we have demonstrated in the previous section is that that if an image can be expressed in terms of a spline basis then it can be reconstructed on our image synthesis hardware. In the remainder of this section we would like to show by referring to the literature on multiresolution analysis, some of which is very recent, that this can be a very powerful technique.

4.1 Wavelets and Spectral Analysis

This is not a tutorial on wavelets: we simply wish to show the applicability of the field to our area of application. If we incidentally motivate readers unfamiliar with wavelets to study the subject then one of the numerous introductions that now exist can be consulted (even in the popular literature, see [7]). Particularly appropriate for our area of interest are [21, 5], general introductions can be found in [8, 26, 9] while useful collections of recent papers can be found in [25, 6, 11] ([6] contains a extensive bibliography).

When the spectrum of a signal is to be obtained by Fourier analysis then all information about the signal, past and future, $-\infty$ to $+\infty$ in all dimensions, must be available. If the signal is altered in some small neighbourhood of a point then the whole Fourier spectrum is affected. Considered another way, if we know that a signal is exactly located as an impulse $\delta(t - t_0)$ at t_0 then its spectrum is spread out over the whole frequency domain as $e^{-it_0\omega}$. Alternatively if we have an exact frequency then the location of the pure sinusoid is indeterminate. Wavelets offer a general compromise in this uncertainty relation, instead of basis functions with infinite support like sines and cosines they use bases with compact support, hence the terms "wavelets" or "ondelettes". An analogy are the notes used to score music: they specify a particular tone at a specified time with various scales of duration.

An early example was the Gabor transform that offered the theoretically optimal combination of location in frequency and space for a fixed size window of uncertainty. Wavelets have a lower simultaneous accuracy in the two domains but have other advantages over early approaches. They automatically zoom in to a high frequency and locate it more accurately and they zoom out to a wide window to analyse low frequencies more accurately. Another characteristic of wavelets is that there is not a unique set, in fact there are infinitely many. We shall consider spline wavelets since they are useful for image compression and are suited to our decompression system.

4.2 Multiresolution Analysis

Wavelet bases are often defined from the data produced by a multiresolution analysis.

A signal $f(x)$ which is "well behaved" in a precise sense ($\in L^2$) can be approximated at various resolutions 2^j, $j \in \mathcal{Z}$. The approximation is achieved by a projection operator P_j which projects the function to a projection space V_j which has a Riesz basis $2^{j/2}\phi(2^j x - k)$ where $k \in \mathcal{Z}$ and $\phi(x) \in L^2$. In this way we obtain a nested sequence of subspaces V_j such that $\ldots \subset V_0 \subset V_1 \subset \ldots \subset V_j \subset V_{j+1} \subset \ldots \subset L^2(\mathcal{R}^d)$. The function ϕ is known as a *scaling function* or *father wavelet*.

The difference information between two approximations of $f(x)$, $P_j f \in V_j$ and $P_{j+1} f \in V_{j+1}$ is given by the orthogonal projection $Q_j f$ onto the complement W_j of V_j in V_{j+1}. The spaces W_j contain the difference between the information at one resolution and the extra information at a higher resolution in a multiresolution analysis of the function. So we have:

$$V_j \oplus W_j \quad = \quad V_{j+1}$$

$$\begin{aligned} W_j &\perp V_j \\ Q_j f &= P_{j+1} f - P_j f \end{aligned}$$

These spaces W_j are spanned by the wavelets ψ, which is also known as the *mother wavelet* or *analysing wavelet*. The ϕ and ψ are related. It can be shown that $\{p_n\}$ and $\{q_n\}$ exist such that:

$$\begin{aligned} \phi(x) &= \sum_k p_k \phi(2x-k) \\ \psi(x) &= \sum_k q_k \phi(2x-k) \end{aligned}$$

The ϕ are used for approximations and the ψ for analysing the errors.

Multiresolution analysis can be done in terms of orthogonal or non-orthogonal basis functions. The former are fairly well understood [21]. The most common operations in image analysis however cannot be cast into the orthogonal wavelet framework. Orthogonal wavelets are complex non-symmetric functions whereas "nice" functions generate non-orthogonal wavelets. For this reason we will propose an investigation into non-orthogonal wavelets and multiresolution analysis [14, 15]; see below.

Spline Wavelets

A non-orthogonal class of wavelets can be obtained from the cardinal spline functions. In fact typical examples of the scaling functions ϕ introduced above are the cardinal splines illustrated in Figure 9. A cardinal spline is a polynomial spline with equally spaced simple knots [5, 28, 29]. The advantages of splines are:

1. The standard basis functions are B-splines that are all convolutions of the unit pulse (the zero order spline in Figure 9).
2. Smooth functions with compact support.
3. Simple analytic forms that are easy to compute and manipulate.

4.3 Wavelet Compression of Images

Many applications of wavelets to image compression have appeared [14, 13, 31, 2, 12]. Some of the most interesting developments have been the papers by Mallat and colleagues who have investigated the "adaptive" compression of images where the compression depends on the detection of important features (edges) and building the compression around these [22, 16, 23]. That this accords with human perceptual predilections can be seen from the ease with which people recognize line drawings of objects which only show the edge features. These methods were able to compress the same image as we used in our example (Figure 8) by factors of between 40 and 100 (≈ 0.081 bits per pixel)..

There is a conjecture by David Marr [24] that if one detects all edges at all the scales then one can reconstruct the image with this information. Wavelets finally allow one to perform this kind of analysis. In [16] on *second generation* image coding the point is made that the same features are called "edges" at one resolution and "textures" at another. Thus if we stop at a particular resolution level the rest of the information that remains (the error term) can be called textures and one can use special encoding methods for those, since the human visual system is not so concerned about the locality of texture elements.

5 Conclusion

We have shown how image reconstruction on a simple polynomial basis can be performed at very high speed using the in-house hardware developed at the CWI. The reconstruction time depends *not* on the spacing of the knots in the splines (the lengths of the interpolation spans) but only on the number of knots. An image can be decompressed even at video rates provided that the number of knots are less than the number of pixels to be generated (by some fixed overhead per scanline).

This demonstration now opens the way for using this hardware to reconstruct images that have been coded with a *wavelet transform*. The wavelet transform is a multiresolution description of the image that can be decoded to yield more and more accurate reconstructions of an image. The transform also precisely locates high-frequency features in space and low-frequency signals in the frequency domain. In fact it is argued [12] that wavelet transforms perform better than the discrete cosine transform advocated by the JPEG standard, it fits in better with human perceptual aptitudes and is a more compact coding.

It is interesting to observe that when Phong shaded images are synthesized with this architecture, that is, when it is used as it was intended, one gets a very efficient encoding of the images in terms of the quadratic[6] spline curve of Figure 3. Moreover the splines are exactly aligned with the edges of the (polygonized) contours in the image. A very efficient single resolution encoding!

It is clear that this use of the architecture for decoding wavelet compressed images will be a very useful area for future research. It is apparent that non-orthogonal wavelets seem to be particularly promising. One major area will be adaptive or "intelligent" coding of images that make use of the techniques developed in low level computer vision. One would detect features which are "interesting" for human perception and optimize the coding of these. Only then can the very high compression ratios (¿ 100) looked for be achieved.

Eventually the system might be used for speech decoding [33, 30], finite element analysis and not just for image decoding. Finally it may be hoped that the investigation into wavelet image coding techniques may in their turn provide useful spin-offs for image synthesis, one could think of new ways to do adaptive anti-aliasing and of representing textures.

[6] Anti-aliasing produces some complications, notably small pieces approximated by cubic polynomials.

References

[1] A. Akman, P. J. W. ten Hagen, and A. A. M. Kuijk. A vector-like architecture for raster graphics. In W. Strasser and A. A. M. Kuijk, editors, *Advances in Computer Graphics Hardware, II*, pages 137–154. Springer-Verlag, Berlin, 1988.

[2] A. C. Bovik, N. Gopal, T. Emmoth, and A. Restrepo. Localized measurement of emergent image frequencies by Gabor wavelets. *IEEE Trans. Information Theory*, 38(2):691–712, March 1992. Special Issue on Wavelet Transforms and Multiresolution Signal Analysis.

[3] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Comm.*, 31:532–540, 1983.

[4] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. on Graphics*, 2(4):217–236, October 1983.

[5] C. K. Chui. *An Introduction to Wavelets*. Wavelet Analysis and Its Applications. Academic Press, Boston, 1992.

[6] C. K. Chui, editor. *Wavelets: A Tutorial in Theory and Applications*. Wavelet Analysis and Its Applications. Academic Press, Boston, 1992.

[7] M. A. Cody. The fast wavelet transform. *Dr. Dobb's Journal*, 17(4):16–28, 100–101, April 1992.

[8] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure and Appl. Math.*, 41:909–996, 1988.

[9] I. Daubechies. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Series in Applied Mathematics*. SIAM, 1992.

[10] I. Daubechies, S. Mallat, and A. S. Willsky. Introduction to the special issue on wavelet transforms and multiresolution signal analysis. *IEEE Trans. Information Theory*, 38(2):529–531, March 1992.

[11] I. Daubechies, S. Mallat, and A. S. Willsky, editors. *Special Issue on Wavelet Transforms and Multiresolution Signal Analysis*, volume 38 of *IEEE Trans. Information Theory*. IEEE, March 1992. Part II of issue number 2.

[12] P. Desarte, B. Macq, and D. T. M. Slock. Signal-adapted multiresolution transform for image coding. *IEEE Trans. Information Theory*, 38(2):897–904, March 1992. Special Issue on Wavelet Transforms and Multiresolution Signal Analysis.

[13] R. A. DeVore, B. Jawerth, and B. J. Lucier. Image compression through wavelet transform coding. *IEEE Trans. Information Theory*, 38(2):719–746, March 1992. Special Issue on Wavelet Transforms and Multiresolution Signal Analysis.

[14] J.-C. Feauveau. Nonorthogonal multiresolution analysis using wavelets. In Chui [6], pages 153–178.

[15] H. G. Feichtinger and K. Gröchenig. Non-orthogonal wavelet and Gabor expansions, and group representations. In Ruskai [25], pages 353–375.

[16] J. Froment and S. Mallat. Second generation compact image coding with wavelets. In Chui [6], pages 655–678.

[17] M. A. Guravage, E. H. Blake, and A. A. M. Kuijk. Xinposse: Structural simulation for graphics hardware. In A. Kaufman, editor, *Advances in Computer Graphics Hardware, VI*. Springer-Verlag, Berlin, 1992. To Appear.

[18] J. A. K. S. Jayasinghe, G. Karagiannis, F. Moelaert El-Hadidy, O. E. Herrmann, and J. Smit. Two-level pipelined systolic array graphics engine. *IEEE Journal of Solid-State Circuits*, 26(3):229–236, 1991. Revised version of paper by the same title in Proceedings IEEE 1990 Custom Integrated Circuits Conference, Boston, Massachusetts. pp. 17.2.1-17.2.4.

[19] J. A. K. S. Jayasinghe, A. A. M. Kuijk, and L. Spaanenburg. A display controller for an object-level frame store system. In A. A. M. Kuijk, editor, *Advances in Computer Graphics Hardware, III*, pages 141–170. Springer-Verlag, Berlin, 1990.

[20] A. A. M. Kuijk and E. H. Blake. Faster phong shading via angular interpolation. *Computer Graphics Forum*, 8(4), 1989. Please note that on p. 321 the definitions of a and b should be swapped.

[21] S. Mallat. A theory of multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 1989.

[22] S. Mallat and W. L. Hwang. Sigularity detection and processing with wavelets. *IEEE Trans. Information Theory*, 38(2):617–643, March 1992. Special Issue on Wavelet Transforms and Multiresolution Signal Analysis.

[23] S. Mallat and S. Zhong. Wavelet transform maxima and multiscale edges. In Ruskai [25], pages 67–104.

[24] D. Marr. *Vision*. W. H. Freeman, 1982.

[25] M. B. Ruskai, editor. *Wavelets and Their Applications*. Jones and Bartlett, Boston, 1992.

[26] G. Strang. Wavelets and dilation equations: A brief introduciton. *SIAM Review*, 31:614–627, 1989.

[27] P. J. W. ten Hagen, A. A. M. Kuijk, and C. G. Trienekens. Display architecture for vlsi-based graphics workstations. In W. Strasser, editor, *Advances in Computer Graphics Hardware, I*. Springer-Verlag, Berlin, 1987.

[28] M. Unser and A. Aldroubi. Polynomial splines and wavelets—a signal processing perspective. In Chui [6], pages 91–122.

[29] M. Unser, A. Aldroubi, and M. Eden. Fast B-spline transforms for continous image representation and interpolation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(3):277–285, 1991.

[30] M. V. Wickerhauser. Acoustic signal compression with wavelet packets. In Chui [6], pages 679–700.

[31] R. Wilson, A. D. Calway, and E. R. S. Pearson. A generalized wavelet transformm for Fourier analysis: The multiresolution Fourier transform and its application to image and audio signal analysis. *IEEE Trans. Information Theory*, 38(2):674–690, March 1992. Special Issue on Wavelet Transforms and Multiresolution Signal Analysis.

[32] A. P. Witkin. Scale-space filtering. In *Int. Joint Conf.on Artificial Intelligence*, pages 1019–1022, 1983.

[33] X. Yang, K. Wang, and S. A. Shamma. Auditory representation of acoustic signals. *IEEE Trans. Information Theory*, 38(2):824–839, March 1992. Special Issue on Wavelet Transforms and Multiresolution Signal Analysis.

Figure 7: Image on Parabola Basis, Steps 3–5.
For description see the caption of Figure 6.

Figure 8: Image on Parabola Basis, Steps 6–8.
For description see the caption of Figure 6.

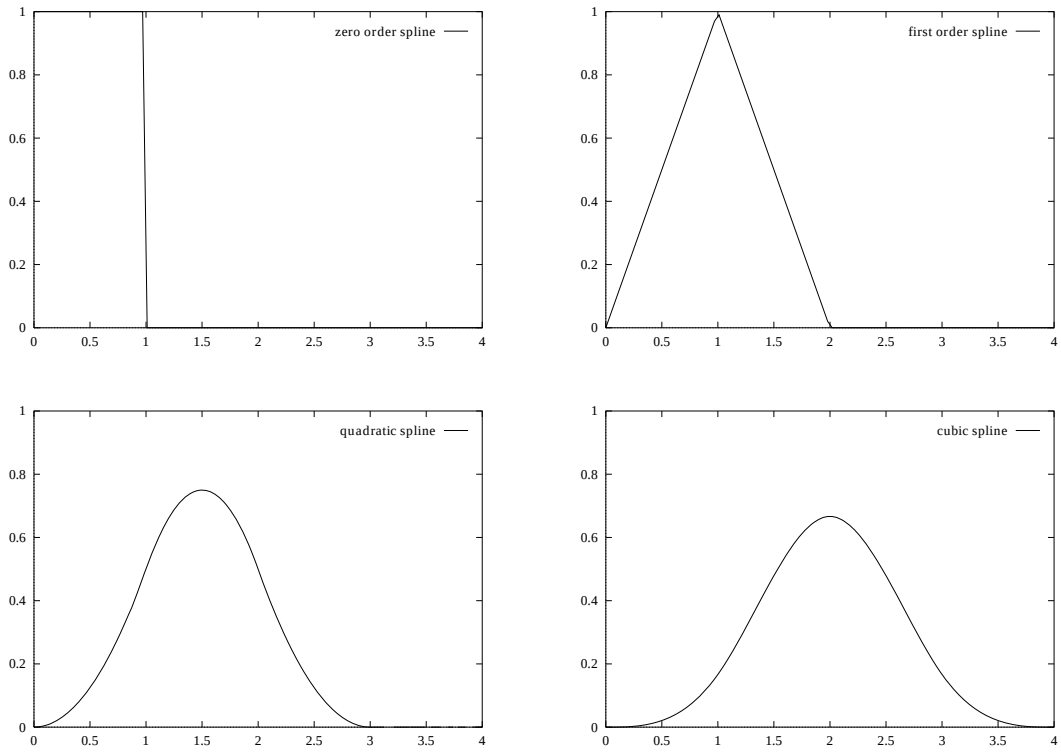

Figure 9: Examples of Cardinal B-Splines