Super Resolution Volume Rendering Hardware

Marco Bosma, Jaap Smit and Jeroen Terwisscha van Scheltinga

University of Twente
Department of Electrical Engineering EF9250
POBox 217, 7500AE Enschede, The Netherlands
e–mail: jaap@nt.el.utwente.nl

Abstract

The resolution obtained in volume rendering is greatly increased over known methods through the introduction of super resolution techniques which make it possible to enlarge the view of the dataset without the introduction of unnecessary positional, gradient and opacity errors. In this paper our "Super Resolution" technique will be introduced along with a corresponding hardware design.

1 Introduction

Renderings of a 3D volume using volume rendering techniques get blurred when an inappropriate color is rendered at the sample position, which occurs when the opacity at the sample position is incorrect or when the local volume gradient, derived from the gradient calculation, is incorrect.

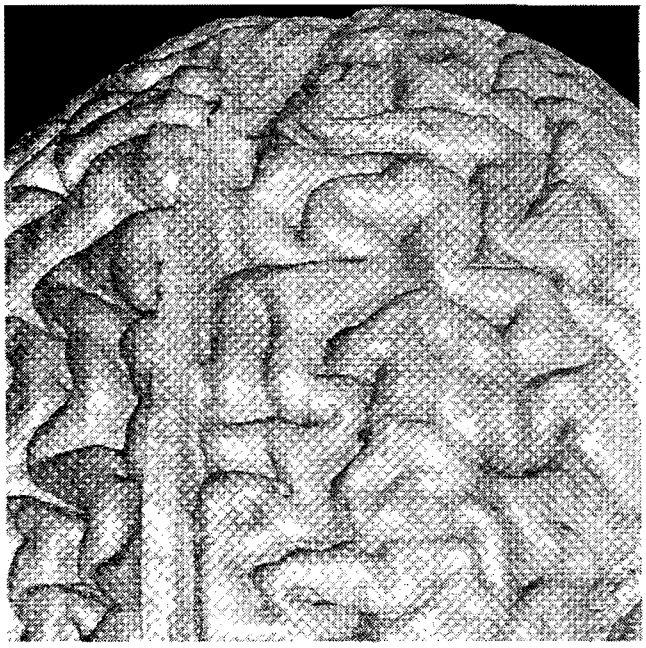


Figure 1. *Brain image generated using the super resolution technique.*

Very high resolution images, like the brain image shown in Figure 1, can be achieved however when color and opacity are correct at all locations on the rays along which the (brain–)image is observed.

The software implementation of the super resolution technique made it possible for us to review the existing algorithms and methods, as we could visualize the errors made. In this paper we present 2D analogies of the 3D visualization process, to show the distinction between the methods used in the literature, substituting contrast for opacity, and cross sections for full 3D views.

Figure 2 shows the reconstruction from sample values of the original circular object, shown in figure 2(a), sampled at a 16 x 16 grid shown in Figure 2(b), using two approaches. The first approach first applies a threshold to calculate the boundary of the object, shown in figure 2(c), and then uses a (linear–) interpolation filter to obtain boundary values at arbitrary locations in the plane as shown in Figure 2(d). Figure 2(e) shows the effect of the approach in which (pre–) calculated opacity values [1] are used to compute opacity values at the sample location, on the rendition of a cone dataset with slits of 1, 0.5 and 0.25 voxel distance, sampled at 32^3 locations.

The problem with this approach is that opacity values should not be (pre–) calculated on voxel positions, instead the grey–value should be interpolated and the non–linear threshold should be applied *after* the interpolation step. Figure 2(f) shows the (linearly–) interpolated grey–values, which can be ideally reconstructed from the samples using Nyquists sampling theorem [5]. The reconstructed contour of the object, shown in Figure 2(g), is very accurate, due to the correct order in which the resampling and the threshold operator are applied. Figure 2(h) shows the effect of this approach when applied to render the cone with slits.

Color and opacity are combined in the composition step to obtain the final image within a volume renderer. A perfect image quality depends hence as well on the ability to calculate the gradient at *arbitrary sample locations* and finally the color from this calculated gradient. In existing volume rendering implementations, a lot of approaches are used to either calculate the color from precomputed colors at voxel locations, or to compute the gradient from precomputed gradients at voxel locations [2],[3]. None of these methods is accurate, as the color should be calculated from the gradient, which in turn should be calculated from the grey–values, without going back to values calculated at voxel positions. The main difference between the approaches commonly used, and the approach proposed in this paper, is related to the improved frequency response of the overall gradient interpolation scheme, which not only has excellent high frequency properties, it is as well free from folding due to the fact that grey–values used in the gradient interpolator are taken from voxel locations just one voxel distance apart, as opposed to gradients calculated with the traditional central gradient algorithm.

The application of correct algorithms to compute the opacity value at sub–voxel locations and gradient algorithms which calculate the color at sub–voxel locations as well, makes an arbitrary magnification of a rendition of a 3D dataset possible without the introduction of unsharpness. This is what we call *Super resolution*.

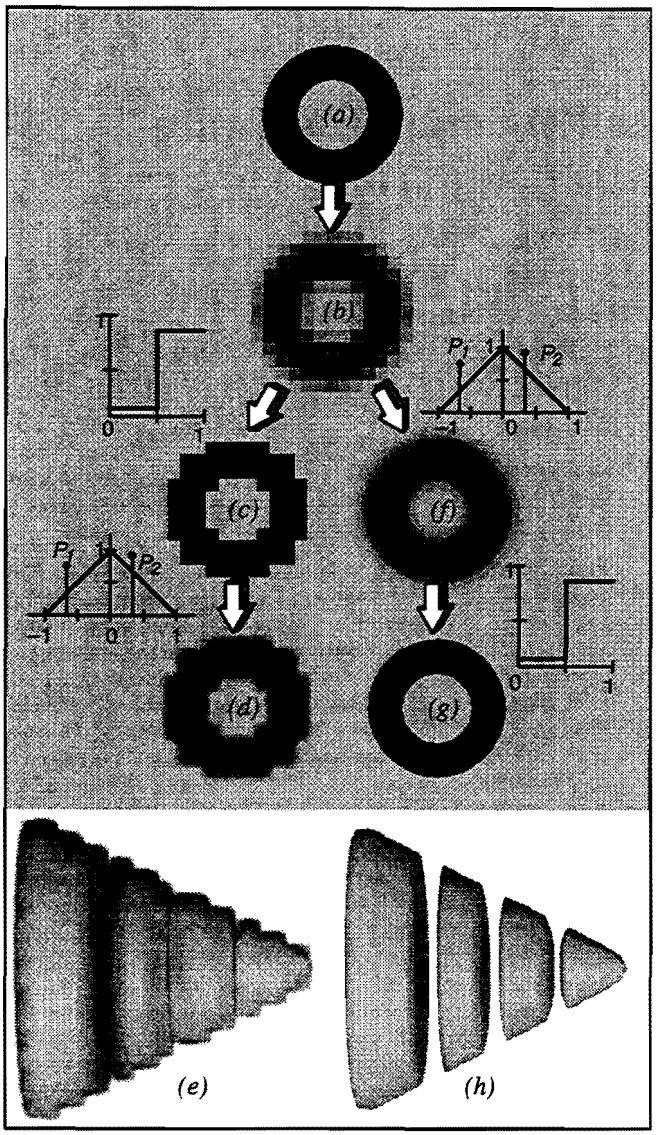


Figure 2. *Accurate opacity calculation.*

Figure 3(b) shows the gradients calculated with our new gradient interpolator at the edge of the reconstructed object shown in figure 3(c). Figure 3(d) shows the object and the corresponding gradients in combination.

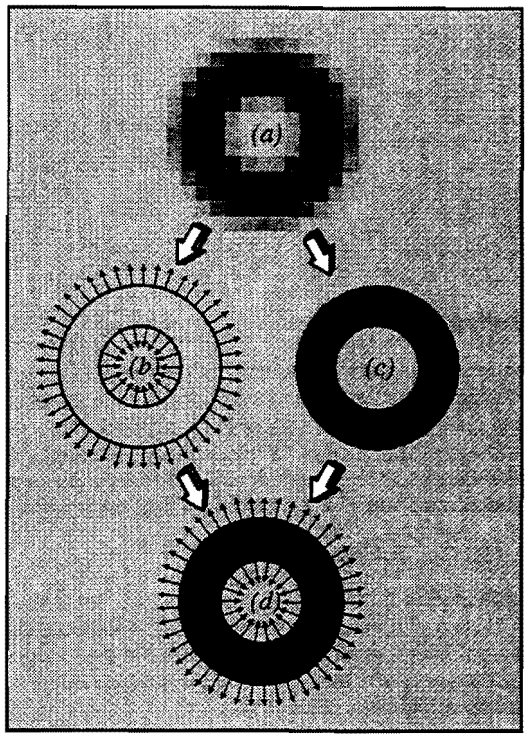
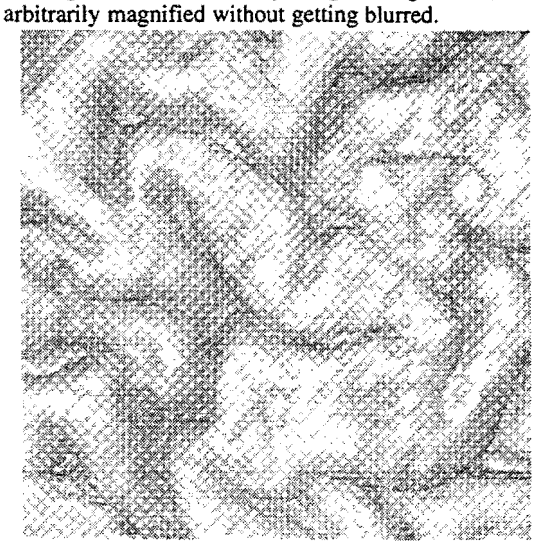


Figure 3. *Accurate gradient calculation.*

In Figures 4(a) and 4(b), renderings of an enlarged brain image are shown. Just the color computation is different in both approaches. The image of Figure 4(a) is rather vague and incorrect due to interpolation of precomputed colors stored at voxel locations. The direct calculation of colors, using Phong shading techniques [4], from gradients at the sample location, based on grey–values at the voxel grid leads to the crisp image of Figure 4(b), which can be arbitrarily magnified without getting blurred.

(a) Colors calculated from precomputed color values stored at voxel–locations



(b) Colors calculated from gradients which are directly calculated at sample locations

Figure 4. *Visualizations using different rendering techniques.*

Remarkable features of the images of Figure 4 (b) and Figure 1 (h), are the sharp edges, the fine details, the true depth and the fact that noise, if present in the dataset, is visualized as appropriately shaded local blots, with approximately the size of the voxel distance in/on the surface of the object visualized. All these properties are preserved when the image is magnified to the limits of the precision of the addressing capabilities of the underlying hardware.

2 The calculation of accurate gradients

The best possible way to calculate the derivative of a function is to differentiate this function. For instance, the differentiation of the function $y(x)=\sin(f \cdot x)$ results in $y'(x)=f \cdot \cos(f \cdot x)$. This implies that the amplitude response of the differentiation function is $|H(f)|=f$, and that its phase is 90 degrees. The problem with volume rendering is however that the function which should be differentiated is only known on the voxel locations. A simple (normalized) subtraction is frequently chosen to be the basis for the calculation of the gradient

$$\nabla v(\mathbf{x}) \equiv (\delta v/\delta x_1, \delta v/\delta x_2, \delta v/\delta x_3)$$

of the voxel values in a 3D dataset.

The six–neighborhood voxel gradient at voxel location $\mathbf{x} = (x_1, x_2, x_3)$ is typically calculated in each of the directions x_1, x_2, x_3, using:

$$\delta v/\delta x_1 \approx (v(x_1+1,x_2,x_3) - v(x_1-1,x_2,x_3))/2\Delta x_1$$

with Δx_1 the voxel distance in x_1.

This calculation has as major disadvantage that the distance between the samples, from which the gradient is calculated, is twice the voxel distance. This effect is reflected in the distance of $2\Delta x_1$ between the discrete value of 0.5 and –0.5 in the impulse response of this operator.

This sampling at a distance of $2\Delta x_1$ is unacceptable, as the sampling theorem [5] states that the highest frequency which can be reconstructed has a period which is twice the sampling distance.

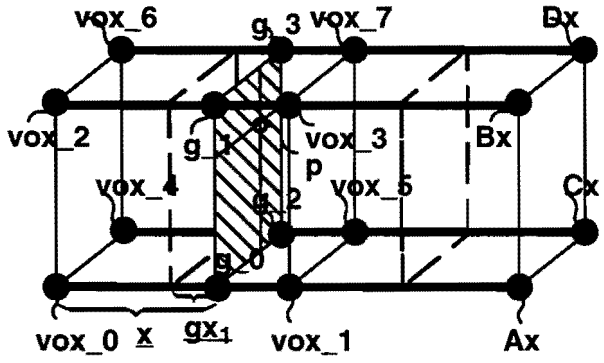


Figure 5. *Voxels and intermediate values used for the calculation of gradients.*

The improved gradient algorithm used in our "super resolution hardware" design is defined for arbitrary non–integral locations in the dataset, as opposed to fixed locations on the voxel grid. The gradient is calculated from a 12 voxel neighborhood of the sample location p, by interpolating four one–dimensional gradients, g_0, g_1, g_2, g_3, calculated on the edges of a pair of unit volumes. These edges extend in the direction of differentiation, shown in bold in Figure 5.

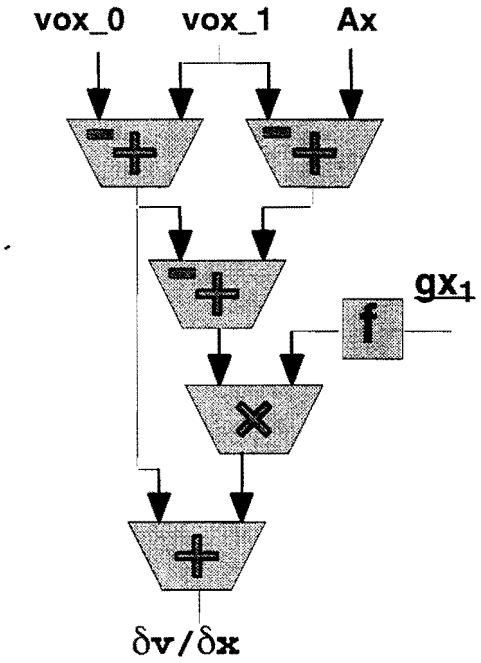


Figure 6. *Gradient calculation unit.*

The hardware used to calculate the gradient at the fractional position $0 \leq gx_1 < \Delta x_1$, in the x_1 direction, from three consecutive voxel locations, vox_0, vox_1 and Ax, is shown in Figure 6.

Figure 7(a) shows the discrete impulse response h (x) in the direction x in which the gradient is taken of the traditional methods which calculate gradients at voxel positions. Figure 7(b) shows the continuous impulse response h (x) designed for use in the super resolution implementation. A simpler, piecewise linear, curve is shown in Figure 7(c).

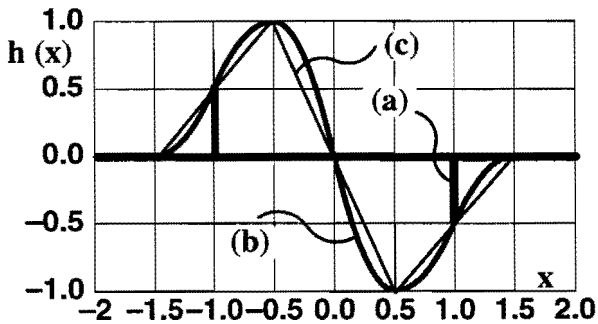


Figure 7. *Impulse responses of three distinct gradient calculation techniques.*

The auxiliary functions f (x), used in the realization of the gradient unit, make h (x) either continuous in its function values and its derivative, this corresponds with Figure 7(b) and 8(b), or just continuous in its function value, this corresponds with Figure 7(c) and 8(c).

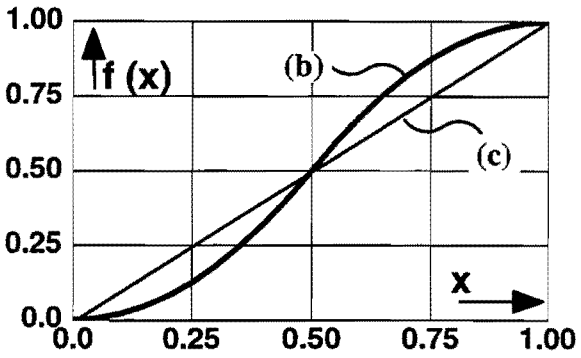


Figure 8. *Two possible functions f(x) to be used in the gradient calculation unit.*

Figure 9 shows the amplitude response for each of the three impulse responses shown in Figure 7, together with the ideal response |H(f)|=f.

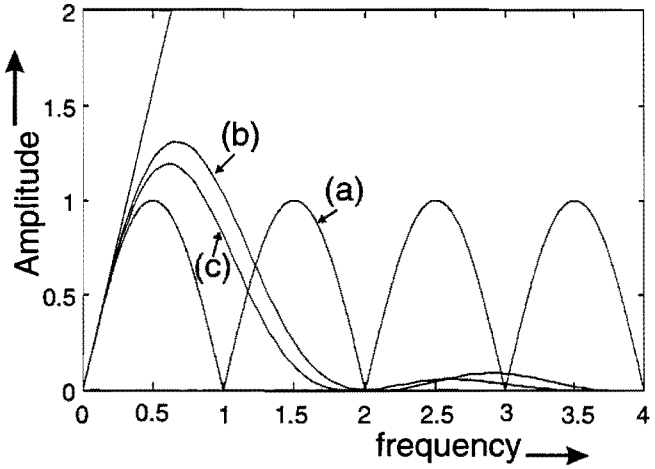


Figure 9. *Amplitude response of the three distinct gradient calculation techniques.*

An implication of the super resolution gradient algorithm is that the gradient calculated is centered at a location which does not coincide with the voxel grid. The fraction g_{x1} used for the calculation of the intermediate gradients g_0 ... g_3, is for this reason either 1/2 voxel location shifted up or down. Eight of the 12 voxels, named vox_0 ... vox_7, used in the calculation of the gradient, are normally used as well to calculate the sample point grey–value from the grey–values in the dataset. The other four voxels, named Ax ... Dx, are either located at lower or at higher addresses. Which side is actually chosen depends on the actual offset, either –0.5 or 0.5, used.

3 Accurate calculation of grey–values

A tri–linear interpolation may be used, using the voxel values vox_0 ... vox_7, to calculate the grey–value at an arbitrary non–integral location in the dataset. The grey–value is ideally registered with the gradient when the procedure for the calculation of the gradient just described is used.

4 Calculation of opacity and color

The application of the opacity function or table is done in the 'super resolution' design at the sample location, i.e. after interpolation of the grey–value from the surrounding 8 voxels when a tri–linear interpolation is used, or from the surrounding 64 voxels when a 3D spline interpolation is used. The color is as well calculated at the sample position. The new gradient algorithm makes this technique, which calculates the color from the gradient and the position of the light sources on sample positions, as opposed to voxel positions, possible. An efficient tabular approach for the calculation of the color from the gradient vector using an on–chip reflectance map as a look–up table is described in [6]. The colors and opacities are used in the composition unit, which uses a front to back technique to calculate the final screen image.

5 Problems with traditional methods

The problem with the traditional method used in [1], based on an intermediate color and opacity dataset, is that the calculation of the color on the sample position from interpolated colors on the voxel grid is not the same as the calculation of the color at the sample position from an accurately generated gradient. The interpolation of colors is a linear process, whereas the calculation of the color from the gradient is a non–linear process. These operations cannot be permuted without the introduction of substantial errors. The errors related to this permutation give rise to blurred edges, as colors of edges at sub–voxel locations are calculated from gradients at voxel locations, where the edge is either not present or not sharp. It may be clear that this effect results in a blurred image.

Another problem with the traditional method used in [1], based on an intermediate color and opacity dataset, is that the calculation of the opacity on the sample position by interpolation of opacities on the voxel grid is not the same as the calculation of the opacity at the sample position from grey–values. The interpolation is a linear process, whereas the calculation of the opacity from the grey–value is a non–linear process. These operations cannot be permuted without the introduction of substantial errors. The errors related to this permutation give rise to vague and notched images.

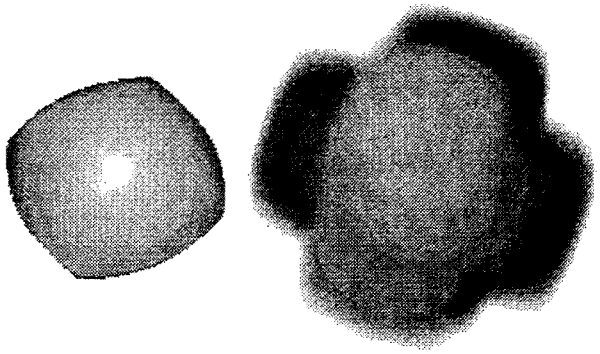


Figure 10. *Observations using (a) the partial volume effect and (b) the incorrect partial opacity effect.*

In Figure 10, this effect is shown for an elementary voxel. In the left image, the opacities are determined correctly at all sample locations in the cube, resulting in a sharp transition between the opaque and the transparent region. In case the opacities at the voxel locations are interpolated, all locations are made more or less opaque, resulting in a vague transition area, as shown in the right image.

Methods that calculate the gradient from linearly interpolated voxel values, like the rendering technique presented by Knittel [7] as a fast alternative, give low quality images due to the fact that differentiation of a piece–wise linear function, resulting from the linear interpolation, will result in piece–wise constant colors.

6 The super resolution algorithm

Super resolution as presented in this paper can only be obtained when the non–linear operations of the assignment of opacities and the computation of the color, from the volume gradient and the position of the light sources, is done at the sample locations. The gradients however should be calculated from the original values at voxel locations. This can be done by a differentiating digital filter, which has an amplitude response which grows linearly with the frequency and a phase which is 90 degrees.

As mentioned, such a filter cannot be derived from a linear interpolator, as the derivative of its triangular impulse response is a piecewise constant function. Better results can be obtained by computing the derivative from voxel values using a function derived through the differentiation of the impulse response of a higher order interpolation function. The alternative, used in this paper, uses a gradient calculator which can calculate gradients at arbitrary locations in the central region of the voxel dataset surrounded by 12 neighboring voxels.

7 Hardware design

Although the presented method results in very high quality images, this method is more expensive as well, compared to the frequently used methods which were primarily optimized for speed. Implementation of the algorithm in software indicated that the rendering process requires about 110 clock cycles per sample point on a RISC CPU for the old method and 164 clock cycles for the new method.

In practical cases, this results in an average total rendering time of 5.5 seconds for super resolution images of size 750x750, like the ones shown in this paper, on an HP735 workstation. These images where generated using multiple depth sample locations per elementary voxel cube. Special hardware, which can process one sample point in one clock cycle, will provide a speed improvement of a factor 100 to 150. This means that volume rendering of sparse datasets can be done in practical cases at a speed of over 20 images per second, using a single coprocessor chip.

To obtain this high volume rendering speed from one single chip, many parallel multipliers are needed for the gradient calculators and the tri–linear interpolator used to calculate grey–values. This means that much effort has to be spent to optimize the multipliers in terms of power dissipation and size.

7.1 The gradient calculation unit

An important aspect of the improved image quality of the method presented in this paper, is the improved calculation of the gradient. Figure 5 shows how the x–direction component of the gradient vector can be determined at the sample location p. This sample point is surrounded by the cube vox_0 ... vox_7, which is needed for the opacity calculation as well. The four voxels Ax ... Dx have to be fetched specifically for the calculation of the x direction component of the gradient vector.

A special gradient calculation unit, shown in Figure 6, is used to calculate the intermediate gradient values in x_i at the locations g_0, g_1, g_2 and g_3 on the bold edges, after which an ordinary 2D interpolator is used to find the desired gradient value g_x at an arbitrary location somewhere in the center of the 12 voxel neighborhood.

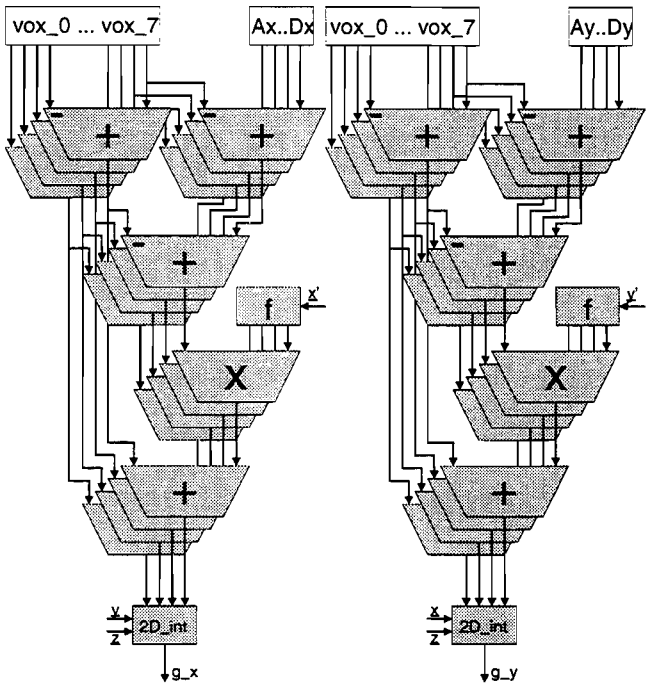


Figure 11. *The 3D gradient calculation units for the x– and y–direction.*

In a similar way, another eight gradient units are needed to calculate g_y and g_z at sample location p. In Figure 11, the datapath for the calculation of g_x and g_y, containing eight gradient units and two 2D interpolators, as used in the full 3D gradient calculation unit, is depicted.

7.2 The opacity, shading and composition units

Using the sample location grey–values and gradients, the value of the corresponding screen pixel has to be calculated. To do so, first the opacity and color have to be calculated at the sample location.

The sample location opacity can easily be calculated by applying a look–up table on the sample location grey–value.

As mentioned, a special shading/coloring unit, can be used to calculate the sample location color based on the sample location gradient vector.

A composition unit is used to calculate the color of the screen pixel based on the sample color and opacity.

In Figure 12, a block scheme for calculating the pixel color, based on the sample location voxel value (v(i,j,k)) and the sample location gradient vector (g_x(i,j,k), g_y(i,j,k), g_z(i,j,k)) as used in the super resolution hardware, is depicted.

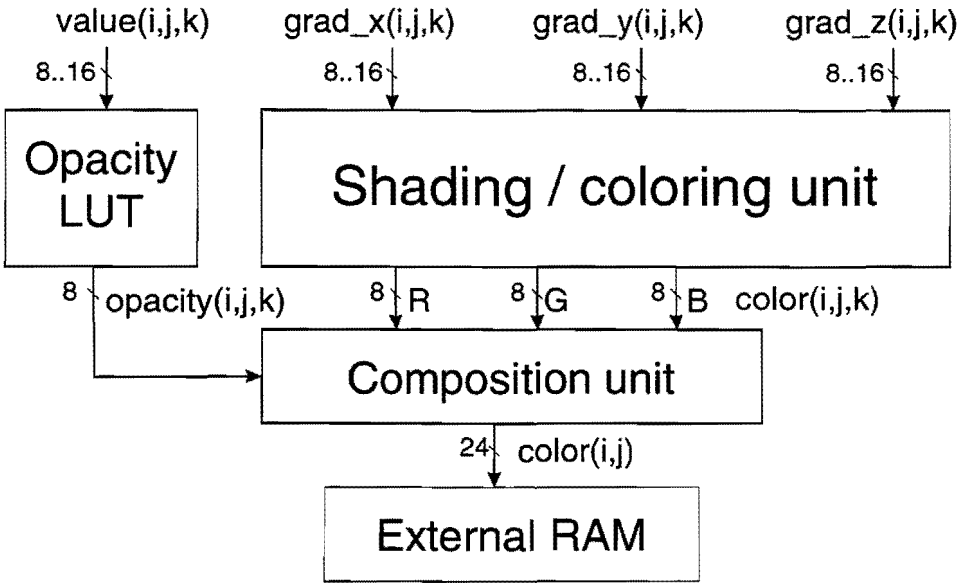


Figure 12. *Units for calculation of opacity and color and the composition of the final image.*

8 Conclusions

A new rendering technique, called *"Super Resolution Volume Rendering"*, which provides renditions which are accurate and sharp within sub–voxel distances has been introduced. Moreover, the design of a volume rendering accelerator, able to generate high resolution images, was presented. The Super Resolution technique makes it possible obtain an arbitrary enlargement of the rendition of a dataset without the introduction of classical errors.

One of the observations made is that linear and non–linear operations should not be permuted to achieve a low computational complexity. Moreover an improved gradient calculation scheme is introduced, which calculates accurate gradients even at high spatial frequencies, without noticeable folding artefacts as opposed to previous methods.

The presented hardware design is able to generate *high quality images* with sharp sub–voxel edges, which are accurately shaded. Noise, if present in the dataset, is visualized as a hammer scale effect on the surface. This noise can be eliminated by applying an additional pre–filtering step on the dataset, which results in extremely sharp and smooth images like the one shown in Figure 1.

CT-scan of engine block!

9 References

[1] M. Levoy, "Display of surfaces from volume data", *IEEE Computer Graphics and Applications*, 8(3), 1988, pp. 29–37.

[2] D. S. Goodsell et al., "Rendering volumetric data in molecular systems", *Journal of Molecular Graphics*, March 1989, pp. 35–36, 41–47.

[3] K. J. Zuiderveld, *Visualization of Multimodality Medical Volume Data using Object–Oriented Methods*, Koninklijke Bibliotheek, Den Haag, 1995, ISBN 90–393–0687–7.

[4] B. T. Phong, "Illumination for computer generated pictures", *Communications of the ACM*, 18(6), June 1975, pp. 311–317.

[5] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice Hall International Editions, 1975, ISBN 0–13–214107–8 01.

[6] J. Terwisscha van Scheltinga, J. Smit and M. Bosma, "Design of an On–Chip Reflectance Map", *Proceedings of the tenth Eurographics Workshop on Graphics Hardware*, 1995.

[7] G. Knittel, "A Scalable Architecture for Volume Rendering", *Proceedings of the ninth Eurographics Workshop on Graphics Hardware*, Sept 12–13, 1994, pp. 58–69.