Adaptive Interpolation

Alastair Reed and Ron Cotner,
Cymbolic Sciences International, Richmond, BC, Canada

Abstract

A basic requirement in any electronic imaging system, is the ability to interpolate image data by resampling the scanned input image. This is particularly true when working with relatively low resolution images on a desktop system, for output on a high resolution color transparency film recorder such as the LightJet 2000.

A major problem with any interpolation algorithm which produces a sharp result is that the optimum sharpness of the resultant image is dependent on local image content. The sharpness of the interpolated image therefore needs to be altered depending on local image content.

We are proposing a method which provides a continuum of interpolated images from relatively sharp at one end to relatively smooth at the other end. A user has the ability to select any degree of sharpness or smoothness in the interpolated output image, within the limits of the full sharpness and full smoothness results. In addition, a user can specify sharpness depending on local image contrast and density.

Introduction

This paper describes a method of adaptively processing a digital input image into an interpolated digital output image, by a method which depends on local image content.

Interpolation to produce a sharp resultant image can be obtained by fitting a cubic or higher polynomial to the input image data. A preferable method of producing a better visual sharp result is to use a digital filter design to obtain weighting coefficients for the input image data, as described in the Overview of Filter Design section. However any interpolation algorithm which produces a sharp result suffers from two major potential problems:

a) Objectionable 'ringing', overshoot and undershoot will occur on high contrast edges in the image. Some degree of overshoot and undershoot is required on lower contrast edges to produce a visually pleasing result, but this will produce objectionable 'ringing' on a high contrast edge. Thus a method of adapting the interpolation algorithm depending on local image contrast is required.

b) The enhancement of unwanted image content such as noise in the shadows due to film grain. Once again some sharpening is required in highlight and midtone areas to produce a visually pleasing result, but this can lead to objectionable graininess in the shadows, and a method of adapting the interpolation algorithm depending on local image density is required.

Methods by which different interpolation algorithms can be used during an interpolation process have been suggested. For example, Fujita et al[1] suggest a method of manually switching between different interpolation algorithms depending on the kind of image to be interpolated. Van Nostrand[2] describes a method of automatically switching interpolation algorithm depending on local image content. With this method, certain corrective measures need to be taken in order to compensate for a sudden switch.

We are proposing a method which provides a continuum of interpolated images from relatively sharp at one end to relatively smooth at the other end. A user has the ability to select any degree of sharpness or smoothness in the interpolated output image, within the limits of the full sharpness and full smoothness results. In addition, a user can specify sharpness depending on local image contrast and density. The amount of overshoot or 'white fringe' can be controlled separately from undershoot or 'black fringe'. Each color can also be controlled separately, if for example the blue channel is the noisiest more smoothing could be applied to the blue channel.

Implementation

For each output pixel, four values are calculated

1) a sharp resultant pixel - P_{sharp}
2) a smooth resultant pixel - P_{smooth}
3) local image contrast - C (scaled between -1 and 1)
4) local image density - D (scaled between 0 and 1)

which are then mixed as shown in Figure 1.

The intermediate output image pixel P_{int} is:

$$P_{int} = P_{sharp} * C_s + (1-C_s)P_{smooth}$$

and the final output image pixel P_{final} is:

$$P_{final} = P_{int} * D_s + (1-D_s)P_{smooth}$$

C_s and D_s are controlled by the user creating look-up tables for the required sharpness as shown in Figure 2 and Figure 3. In Figure 2, a table for smoothing at high contrast and sharpening at medium and low contrast is illustrated. In Figure 3, a table for smoothing at high density and sharpening at medium and low density is shown.

For the sharp result, the adaptive algorithm dictates that we enhance the image *as* we interpolate, that is, the pixels are interpolated and sharpened *simultaneously*. Most

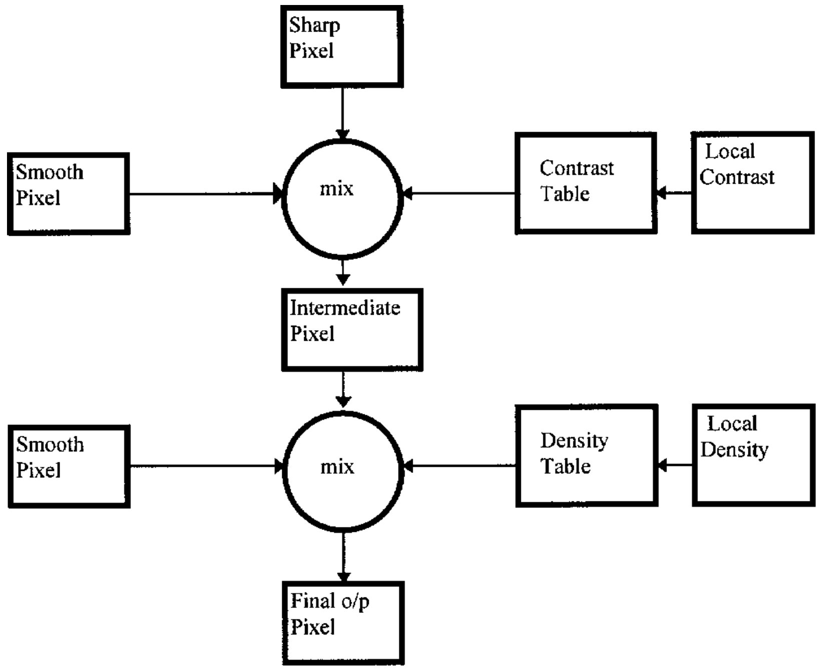

Figure 1. Mixing of Sharp and Smooth Pixels

Figure 2. Sharpness versus Contrast

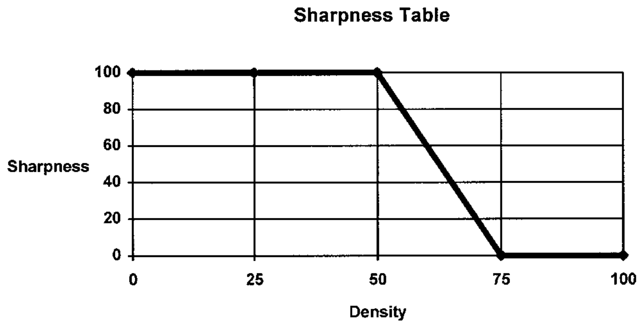

Figure 3. Sharpness versus Density

spatial interpolation schemes such as those based on bicubic, bilinear and polynomial methods are either low pass by nature(as a result of averaging) or produce minimal sharpening and contrast enhancement. A new spatial interpolation method was developed based on design in the frequency domain. This method produces zero-phase, approximately circularly symmetric, even or odd order two dimensional (2D) filters. The scheme is a frequency transformation method described by Lim that was modified and enhanced to allow design of even order filters and contour preshaping to allow interpolation in the spatial domain. It can be shown that a zero-phase circularly symmetric frequency response will produce zero-phase circularly symmetric spatial coefficients. The filter coefficient matrix is then subsampled and convolved as required to give the desired interpolated and enhanced image output pixel.

Overview of Filter Design

The frequency transformation method described by Lim works well for our application (please see reference 3 for a detailed discussion). The use of zero phase filtering minimizes phase distortion in the interpolated image by preserving the positioning of different spectral components in the passband and the transisitonal bands. A circularly symmetric frequency response eliminates directional amplitude biasing in the frequency domain, and circularly symmetric coefficients can lead to less mathematical computation with filter implementation. The frequency transformation method generates a 2D frequency response that is increasingly circularly symmetric as ω decreases. This is particularly useful in designing filters with low cutoff frequencies such as our application requires.

The basic idea of frequency transformation is shown in Figure 4. The process begins with the design of a zero phase 1D FIR filter prototype which represents a 1D view of the desired 2D response. We chose the McClellan-Parks FIR digital filter design algorithm because of the simplicity involved in specifying an arbitrary magnitude frequency response. Although the linear phase filter algorithm generates coefficients for a causal filter, a shift in index by (N-1)/2, N odd, gives coefficients for a zero phase frequency response.

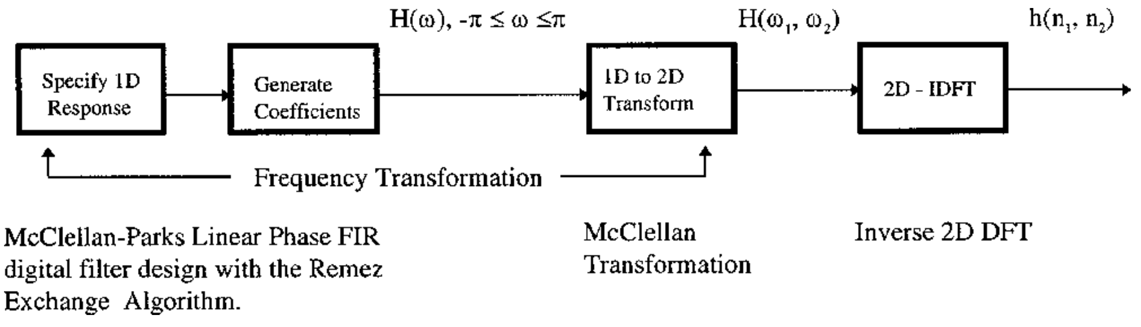

Figure 4. Frequency Transformation Implementation

$H(\omega)$ as required for the McClellan Transformation. In short, these 1D coefficients h(n) are transformed into 2D space as

$H(\omega_1, \omega_2) = H(\omega)_{\cos\omega=T(\omega 1,\, \omega 2)}$ where

$H(\omega) = h(0) = \Sigma n\ 2h(n)*\cos\omega n,\ 1\le n\le (N-1)/2$, N odd

$T(\omega 1, \omega 2) = 0.5(\cos\omega 1 + \cos\omega 2 + \cos\omega 1\omega 2-1)$
(McClellan transformation)

$-\pi\le\omega\le\pi$	1D frequency
$-\pi\le\omega_1\le\pi,\ -\pi\le\omega_2\le\pi$	2D frequency

$H(\omega_1, \omega_2)$ is the desired circularly symmetric zero phase 2D frequency response. The 2D filter coefficients $h(n_1, n_2)$ are then obtained by a 2D inverse DFT taken symmetrically about ω_1 , $\omega_2 = 0$.

This method was presented for N odd. Since physical memory is generally available in powers of 2, coefficient access and address indexing is easier to accomplish for N even. The above algorithm was modified to allow N even by introducing phase shifts corresponding to 1/2 sample in all equations and modifying the expression for $H(\omega)$ appropriately. The symmetric IDFT must also be shifted about ω_1 , $\omega_2 = 0$ by 1/2 sample to give real zero phase coefficients.

Interpolation Filtering

One way of interpolating a 1D bandlimited sequence x(n) by I, an integer, is to insert I-1 zeros between samples of x(n) to form a new sequence s(r) with fourier transform $S(e^{j\omega T/I}) = X(e^{j\omega T})$, where T is the original sampling period[4]. Now, if this new sequence is filtered with an ideal rectangular low pass filter with cutoff at $\omega_0 = \pi / T$, the output will be a scaled and interpolated version of the input sequence. For $T = I*T_1$ where T_1 is the new (shorter) sampling period, the cutoff frequency is at $\omega_0 = \pi / I*T_1$. But for any sampling frequency ω_{s1}, $\omega_{nyq} = \pi / T_1$, so $\omega_0 = \omega_{nyq} / I$. If we normalize with respect to ω_{s1} by setting $\omega_{s1} = 1$, and given that $\omega_{nyq} = \omega_{s1} / 2$, then $\omega_{no} = .5 / I$. This establishes the ideal cutoff frequency for the interpolation filter. Assuming ideal filtering and bandlimiting, the Fourier transform of the (I*N-1) point output sequence in the region $\omega \le \pi\ / T$ is identical to that of the Fourier transform of the (N-1) point input sequence in the region $\omega \le \pi\ / T$.

This argument can be extended to include 2D filters. Using frequency transformation, however, it is much simpler to specify the interpolation filter cutoff as part of the 1D filter design, then apply the 1D to 2D transformation to generate a 2D interpolation filter capable of spatial interpolation.

Filter Requirements

A versatile image interpolating system will support a variety of interpolation factors. Each factor may demand a separate interpolating filter design. For large factors, computation of the 2D filter coefficients is very time consuming; therefore, it is more convenient to design one filter that can handle all interpolation factors. One way to accomplish this is to design the 2D filter for maximum interpolation, then decimate the 2D filter coefficients appropriately. With similar arguments and assumptions used above for interpolating filters, decimating by factor D of a sequence bandlimited to $\omega_{no} = .5 / D$ (normalized) will give a scaled replica of the original Fourier transform in the region $\omega_{no} \le .5$. Applying this to two dimensions, an interpolating 2D filter designed for maximum interpolation factor I_{max} can be decimated by D (in both directions) to give a final interpolation factor of I_{max} / D. Using this method, there is only one 2D filter design that is subsampled and scaled appropriately to give the final interpolation filter, and, the final interpolation factor I_{max} / D need not be an integer.

So far, interpolating 1D filters have been characterized as having magnitude = 1 for $\omega_{no} \le .5 / I$ and 0 otherwise. The restriction of bandwidth eliminates aliasing and the constant passband magnitude preserves the original frequency response in the interpolated one. Altering the passband magnitude, however, will in turn alter the interpolated frequency response. To sharpen *and* interpolate the input image, the passband magnitude must be some form of high pass filter at a minimum. Since the desired result is to sharpen the image, the passband magnitude is contoured to pass the low frequency information but to amplify or emphasize the higher frequencies as shown in Figure 5. This tends to preserve overall brightness of the image while sharpening. The peak normalized frequency and amplitude will determine the degree of sharpness.

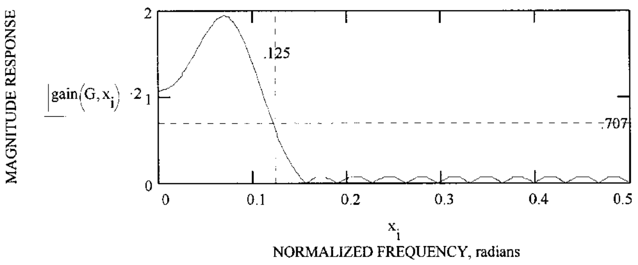

Figure 5. Magnitude Response of a 32 poiint FIR filter pre-designed for interpolation.

Filter Implementation

The real time sharpening interpolator is realized as a 4×4 convolution of the input image with the pre-designed *seed interpolator* decimated by D to give the final interpolation factor $I = I_{max} / D$. To generate the seed interpolator at $I = I_{max}$, the cutoff frequency is placed at $\omega_{no} \leq .5 / I_{max}$. The resulting 2D filter size is then $4I \times 4I$. For $I = 1$(no interpolation), the filter will be decimated by I_{max}. The seed interpolator can be designed from a 4I point 1D filter then transformed to two dimensions. There are, however, two disadvantages to this approach. One, we specify $I_{max} = 256$, which leads to a 1024×1024 point filter design. The computational time for the McClellan Transformation and the 2D - IDFT is exceeding long for such a large filter. Two, the main lobe of the 2D coefficient array, symmetric about ω_1 , $\omega_2 = 0$, is very thin so that most of the coefficients throughout the array are very small numbers relative to the central lobe. From an *intuitive* point of view, it is more desireable to have the energy more evenly spread over the coefficient array. Therefore, a more practical method is shown in figure 6.

Starting with a β point 1D filter design, set $\omega_{no} = .5 / (\beta/4)$. β must be large enough for the FIR filter design to render sufficient detail with the sharpening magnitude response, but small enough to limit computational time and maintain a large central lobe in the spatial filter coefficient array. $\beta < 75$ works well. Next, complete the frequency transformation. The result here is a $\beta \times \beta$ point image sharpening filter coefficient array with cutoff at ω_{no1} , $\omega_{no2} = .5 / (\beta/4)$ designed for maximum interpolation β. To get the final 1024×1024 point unit step response for $I_{max} = 256$, interpolate in both directions by I_{max} / β. Assuming that the $\beta \times \beta$ point image

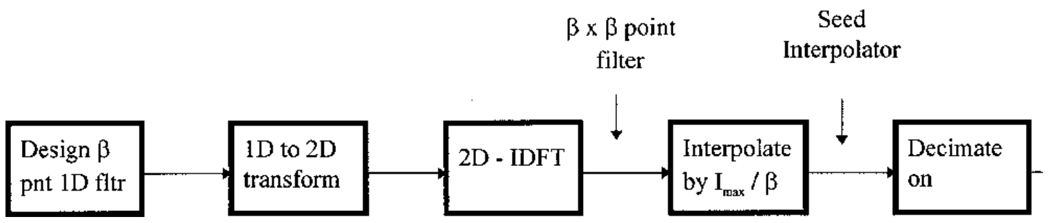

Figure 6. Practical Frequency Transformation of Interpolation Filters

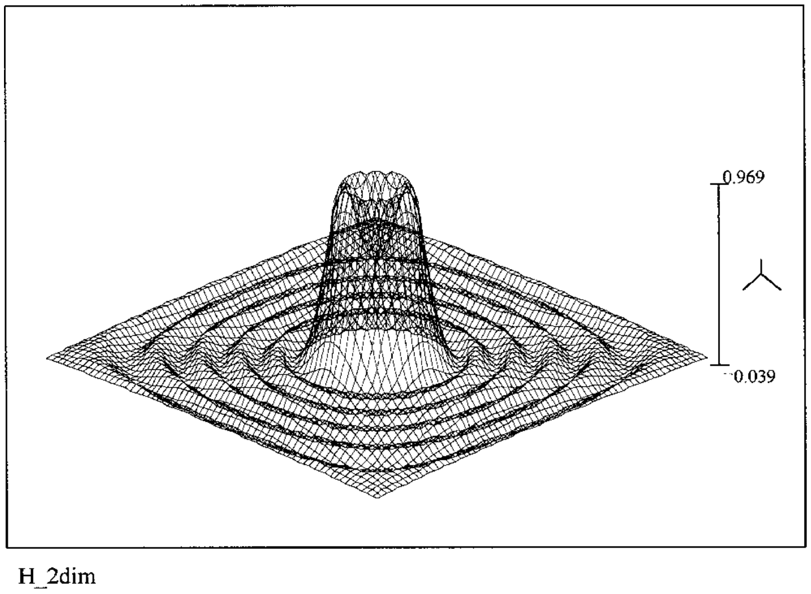

Figure 7. Magnitude Response of a 32x32 point spatioal filter pre-designed for interpolation

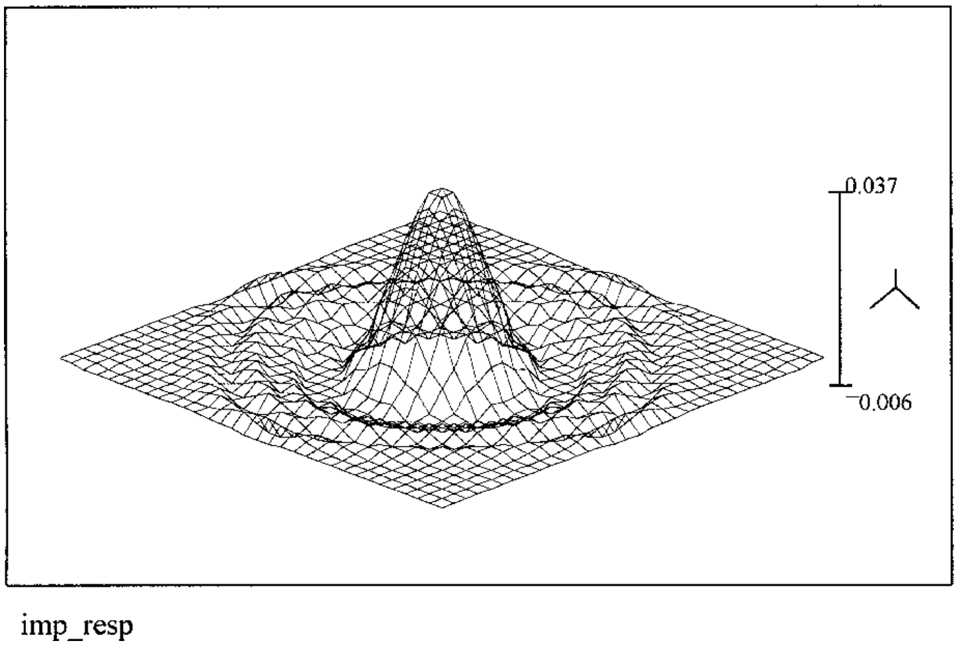

Figure 8. Spatial interpolating filter coeffeicients for the filter of Figure 7

filter is truly bandlimited as required for interpolation, the result will be a 1024 × 1024 point image sharpening filter with cutoff at ω_{no1} , ω_{no2} = .5 / (I_{max} / 4). This filter serves as the seed interpolator. Since the $\beta \times \beta$ point coefficients were interpolated to generate the seed interpolator, the relative size of the central lobe remains the same, but the frequency response and cutoff frequency have been shifted to allow a larger interpolation factor as stated above. The array can stored as a constant, and on demand can easily be decimated to give the final interpolation factor.

Figure 7 illustrates a simple example of a spatial sharpening filter designed for interpolation by 4. The design is based on the 1D prototype on Figure 5. Figure 8 is the final 32×32 spatial interpolating filter coefficient array.

For the smooth result, a similar method was used to produce a low pass filter with no ringing, circular symmetry and zero phase distortion.

Hardware

The adaptive interpolation algorithm is computationally intensive requiring more than 146 million 16 bit multiplies and 44 million adds per second. The memory bandwidth required is on the order of hundreds of megabytes per second. Although a software solution is possible, processing time for typical 3 or 4 color images could take on the order of hours, whereas a hardware solution produces an 8” by 10” image at a plotting resolution of 2000 lines per inch, i.e. 320 Mpixels with 36 bits/pixel in 10 minutes. The algorithm is implemented with several 2D hardware convolvers, DSPs, and FPGAs for address generation, control and other processing functions. Sharp, smooth, contrast and density coefficients are downloaded and stored in DRAM.

The adaptive interpolation algorithm is computationally intensive requiring more than 146 million 16 bit multiplies and 44 million adds per second. The memory bandwidth required is on the order of hundreds of megabytes per second. Although a software solution is possible, processing time for typical 3 or 4 color images could take on the order of hours, whereas a hardware solution produces an 8” by 10” image at a plotting resolution of 2000 lines per inch, i.e. 320 Mpixels with 36 bits/pixel in 10 minutes. The algorithm is implemented with several 2D hardware convolvers, DSPs, and FPGAs for address generation, control and other processing functions. Sharp, smooth, contrast and density coefficients are downloaded and stored in DRAM.

Conclusions

Adaptive interpolation works very successfully, avoiding the limitations of any one *fixed* interpolation algorithm which will always involve compromises . Two common examples are:

a) Sharp interpolation to obtain a pleasing image in low to medium contrast areas can result in objectionable ringing in high contrast areas. The only way to avoid this problem is to adaptively change the interpolation algorithm used to minimise the ringing in high contrast areas, as shown in Figure 2.

b) Sharp interpolation to obtain a pleasing image at low to medium density can result in enhancement of shadow noise in high density areas. By adaptively changing the interpolation algorithm from full sharpness at low and medium density, to full smoothness at high density to reduce shadow noise, as shown in Figure 3, a good result can be obtained over the full tonal range.

An ASIC implementation is being investigated for better performance at less than one quarter the cost.

References

1. F. S. Fujita, H. K. Sugizaki and K. K. Kiyota, US Pat No. 4,468,693, ‘Video Printing Apparatus’.
2. L. Van Nostrand, US Pat No 5,008,752, ‘Digital Image Interpolator with Multiple Interpolation Algorithms’
3. J. S. Lim, ‘Two Dimensional Signal and Image Processing’, pp 218-237, Simon and Schuster, New Jersey, 1990.
4. N. K. Bose, ‘Digital Filtering, Theory and Applications’, North-Holland, Amsterdam, The Netherlands 1985.