Fast Exact Area Image Upsampling with Natural Biquadratic Histosplines

Nicolas Robidoux[1], Adam Turcotte[1], Minglun Gong[2], and Annie Tousignant[1]

[1] Laurentian University, Sudbury ON P3E 2C6, Canada
nrobidoux@cs.laurentian.ca
http://www.cs.laurentian.ca/resampling
[2] Memorial University of Newfoundland, St. John's NL A1C 5S7, Canada

Abstract. Interpreting pixel values as averages over abutting squares mimics the image capture process. Average Matching (AM) exact area resampling involves the construction of a surface with averages given by the pixel values; the surface is then averaged over new pixel areas. AM resampling approximately preserves local averages (error bounds are given). Also, original images are recovered by box filtering when the magnification factor is an integer in both directions. Natural biquadratic histosplines, which satisfy a minimal norm property like bicubic splines, are used to construct the AM surface. Recurrence relations associated with tridiagonal systems allow the computation of tensor B-Spline coefficients at modest cost and their storage in reduced precision with little accuracy loss. Pixel values are then obtained by multiplication by narrow band matrices computed from B-Spline antiderivatives. Tests involving the re-enlargement of images downsampled with box filtering suggest that natural biquadratic histopolation is the best linear upsampling reconstructor.

1 From Point Values to Pixel Averages

Image upsampling is most commonly implemented as a two step process [1]. First, interpolation is used to construct a continuous version of the image: a surface $f(x, y)$ such that

$$f(x_j, y_i) = p_{i,j} \quad \text{(reconstruction)}. \tag{1}$$

Here, $p_{i,j}$ is the pixel value with index (i, j), and (x_j, y_i) is the position of the corresponding pixel. The reconstructed surface is then resampled at the desired rate, that is, the pixel values $P_{I,J}$ of the upsampled image are given by

$$P_{I,J} = f(X_J, Y_I) \quad \text{(sampling)}, \tag{2}$$

where (X_J, Y_I) is the position of the corresponding pixel in the enlarged image.

1.1 Average Matching (AM) Image Resampling

Making the reconstructed light intensity surface have point values matching the pixel values as in Eq. (1) ignores the fact that image sensors count incoming

A. Campilho and M. Kamel (Eds.): ICIAR 2008, LNCS 5112, pp. 85–96, 2008.

photons over small non-overlapping areas, so that pixel values are better interpreted as averages than point values [2]. This is a gross simplification of the image capture process [3]. In addition, raw digital images are usually further processed prior to magnification. It is nonetheless reasonable to expect the average value interpretation to yield better resampling schemes than the point value interpretation [4]. We define average matching (AM) resampling to be exact area resampling in which the pixels of the input image are assumed to be abutting squares, and those of the output image, abutting rectangles [1]. In an AM method, the reconstructed intensity surface is defined by

$$\frac{1}{h^2}\int_{y_i-h/2}^{y_i+h/2}\int_{x_j-h/2}^{x_j+h/2} f(x,y)\,dxdy = p_{i,j} \qquad \text{(reconstruction)}$$

instead of Eq. (1). Here, (x_j, y_i) is the center of the square pixel with index (i, j) and h is the pixel's width and height as well as the horizontal and vertical distance between adjacent pixel centers. With i (resp. j) running from 0 to $m-1$ (resp. $n-1$), it is convenient to set $x_j = j+1/2$ (resp. $y_i = i+1/2$) so that the pixels of the input image have unit sides. Then, an input image with m rows and n columns has width n and height m, dimensions which differ from those usually associated with interpolatory resampling, for which the placement of pixel points right at the boundary of the image is generally understood to imply an image width of $n-1$ and height of $m-1$. Likewise, the pixel values of the upsampled image are given by

$$P_{I,J} = \frac{1}{\Delta X \Delta Y}\int_{Y_I-\Delta Y/2}^{Y_I+\Delta Y/2}\int_{X_J-\Delta X/2}^{X_J+\Delta X/2} f(x,y)\,dxdy \qquad \text{(sampling)}$$

instead of Eq. (2); here, (X_J, Y_I) is the position of the center of the resampled image's pixel with index (I, J), ΔX is the pixel width, and ΔY is the pixel height. For an output image with M rows and N columns, choosing $\Delta X = n/N$ and $\Delta Y = m/M$ makes the implied dimensions of the output image identical to those of the input image. With these conventions, the steps of an AM method are

$$\int_i^{i+1}\int_j^{j+1} f(x,y)\,dxdy = p_{i,j} \qquad \text{(reconstruction)}, \quad (3)$$

$$P_{I,J} = \frac{MN}{mn}\int_{I\frac{m}{M}}^{(I+1)\frac{m}{M}}\int_{J\frac{n}{N}}^{(J+1)\frac{n}{N}} f(x,y)\,dxdy \qquad \text{(sampling)}. \quad (4)$$

1.2 Box Filtering, the Simplest AM Resampler

The simplest and most common AM resampler is box filtering, for which the reconstructed surface is constant over each pixel area; that is, Eq. (3) is satisfied by setting $f(x,y) = p_{i,j}$ over the square $(j\,, j+1)\times(i\,, i+1)$, as in nearest neighbor interpolation. (In box filtering, the values of f at points halfway between two pixel centers are irrelevant because pixel boundaries do not contribute anything to the integrals.) Box filtering is commonly used to downsample images. For example, it is the default GNU Image Manipulation Program (GIMP) downsizing method.

As an upsampling method, however, box filtering is not very popular, giving results similar—often identical—to nearest neighbor interpolation. Nonetheless, it is the "quality" Java Abstract Window Toolkit (AWT) image magnification method [5]: enlargement by pixel replication to a large intermediate image with dimensions equal to the LCMs of those of the input and output images, followed by averaging of the values of pixels which cover each output pixel.

1.3 AM Methods Approximately Preserve Local Averages

AM methods share a very attractive property: The pixel averages of the resampled image approximate those of the original image over corresponding regions. More specifically, let Σ be a subset of $\{0,1,\ldots,M-1\}\times\{0,1,\ldots,N-1\}$ and

$$\Omega = \bigcup_{(I,J)\in\Sigma} \left[J\frac{n}{N},(J+1)\frac{n}{N}\right]\times\left[I\frac{m}{M},(I+1)\frac{m}{M}\right],$$

be the region covered by the pixels with indices in Σ, with area $|\Omega|$ equal to $(|\Sigma|\,mn)/(MN)$, $|\Sigma|$ being the cardinality of Σ. Also let σ be a collection of input pixel indices and ω be the region covered by the corresponding pixels. Because input pixels have unit area, $|\omega| = |\sigma|$. Eq. (3)–(4) imply that

$$\int_\Omega f\,dA = \frac{|\Omega|}{|\Sigma|}\sum_{(I,J)\in\Sigma} P_{I,J}, \qquad \text{and} \quad \int_\omega f\,dA = \frac{|\omega|}{|\sigma|}\sum_{(i,j)\in\sigma} p_{i,j}.$$

Now, suppose that minval, the smallest possible pixel value, is 0. The difference between the respective pixel averages satisfies

$$\begin{aligned}\left|\frac{1}{|\Sigma|}\sum_{(I,J)\in\Sigma} P_{I,J} - \frac{1}{|\sigma|}\sum_{(i,j)\in\sigma} p_{i,j}\right| &\le \frac{1}{|\omega|}\int_{\Omega\Delta\omega}|f|\,dA + \left|1-\frac{|\Omega|}{|\omega|}\right|\left(\frac{1}{|\Sigma|}\sum_{(I,J)\in\Sigma} P_{I,J}\right)\\ &\le \frac{|\Omega\Delta\omega|}{|\omega|}\left(\max|f| + \frac{1}{|\Sigma|}\sum_{(I,J)\in\Sigma} P_{I,J}\right).\end{aligned}$$

If the alignment of the pixels of the input and output images is such that Ω is a union of input pixels, choosing $\omega = \Omega$ makes the two averages identical because the symmetric difference $\Omega\Delta\omega$ is empty. This is the case when Ω covers the entire image. Consequently, pixel averages are globally preserved by AM methods.

We now specifically address local average preservation. When m divides M and n divides N, one can make Ω equal to ω because output pixels are obtained by evenly subdividing input pixels. Consequently, averages over unions of input pixels are exactly preserved when the magnification factor is an integer in both directions. Given Σ, however, it is generally impossible to choose σ so that $\Omega = \omega$. For simplicity, suppose from now on that Ω is a rectangle with k output pixel rows and l columns. It is always possible to choose σ so that ω is a rectangle with boundary at most 1/2 away from the boundary of Ω. With this choice,

$$\left|\frac{1}{|\Sigma|}\sum_{(I,J)\in\Sigma} P_{I,J} - \frac{1}{|\sigma|}\sum_{(i,j)\in\sigma} p_{i,j}\right| \le \frac{k\frac{m}{M}+l\frac{n}{N}+2}{\left(k\frac{m}{M}-1\right)\left(l\frac{n}{N}-1\right)}\left(\max|f| + \frac{1}{|\Sigma|}\sum_{(I,J)\in\Sigma} P_{I,J}\right). \qquad (5)$$

Provided Ω has a bounded aspect ratio, the first factor of this upper bound goes to zero as $|\Sigma|$ increases. Because pixel averages can't be larger than maxval (the largest possible pixel value) and, for a reasonable reconstructor, $|f|$ stays within a small multiple of maxval, the opening statement of this section is established, at least as far as averages over large squarish rectangles are concerned.

If the reconstructed surface $f(x, y)$ stays within the interval $[0, \text{maxval}]$, as is the case for monotone reconstructors, in particular for the nearest neighbor reconstructor which is the basis of box filtering, the last factor of the bound which appears in Eq. (5) is at most $2\,\text{maxval}$. Natural biquadratic histopolation, however, is not monotone: the range of f is generally not contained within the range of pixel values. We conjecture that the worst possible value of f for natural biquadratic histopolation is found at the center of an "infinite" checkerboard with an odd number of rows and columns. Using symmetry, this can be shown to be $9\,\text{maxval}\,/\,4$, so that the last factor of Eq. (5) is conjectured to be bounded by $13\,\text{maxval}\,/\,4$ (provided no clamping occurs; see §1.4). (Crude estimates based on the maximum values of the four, six or nine biquadratic B-spline basis functions with support overlapping an input pixel, together with coefficient estimates which rely on the infinity norm of the inverse of the matrix A discussed below, lead to a rigorous bound on the last factor of Eq. (5) equal to $489\,\text{maxval}\,/\,32$.)

1.4 Box Filtering Is a Left Inverse of Integer AM Upsampling

A well-known property of box filtering—and some implementations of nearest neighbor interpolation, ImageMagick's among them—is that if an image is upsampled by an integer factor in both directions, then downsampled back to the original size, the original image is recovered. In other words, box filter downsampling is a left inverse of box filter upsampling when m divides M and n divides N. Because integrating, and consequently averaging, over several pixels is the same as averaging the pixel averages, this property also holds for all monotone AM methods. With a non-monotone reconstructor, $f(x, y)$ may overshoot maxval or undershoot minval. Although overshoots and undershoots are averaged out somewhat by the sampling step (4), which involves box filtering over the areas of output pixels, some pixel values end up being clamped down to maxval, leading to "average intensity loss," or clamped up to minval, leading to "average intensity gain." Consequently, downsampling with box filtering a clamped integer enlargement back to its original size may not return the original, and the above bounds on pixel averages may not hold. (Rounding to integer pixel values and round off error also contribute to these properties being approximate even for monotone AM methods, but their contributions tend to average to zero, while clamping tends to locally happen in only one of the two possible directions.)

2 Natural Biquadratic Histosplines

C. de Boor [6] introduces parabolic "area matching" splines in the context of histogram smoothing: Given $n+1$ real numbers $x_0 < x_1 < \cdots < x_n$ defining

the bins of n histogram bars, and n numbers $p_0, p_1, \ldots, p_{n-1}$ which define their heights, the natural quadratic histopline is the unique continuously differentiable function $f(x)$ with domain $[x_0, x_n]$ such that $f(x)$ is a quadratic polynomial on each interval $[x_j, x_{j+1}]$, $\frac{1}{x_{j+1}-x_j}\int_{x_j}^{x_{j+1}} f(x)\,dx = p_j$ for every j, and $f'(x_0) = f'(x_n) = 0$. Alternately, it can be defined as the derivative of the natural cubic spline [6] which interpolates the cumulative integral associated with the pixel values, that is, the natural cubic spline with value $\sum_{k=0}^{j-1} p_k$ at x_j [7,8]. Extending this construction to the bivariate situation by tensor product [6] yields the following:

Definition 1. *The* natural biquadratic histospline *surface function is the unique continuous function f with domain $[0,n]\times[0,m]$ such that*

- *$f(x,y)$ is a linear combination of 1, x, y, x^2, xy, y^2, x^2y, xy^2 and x^2y^2 on every input pixel $[j\,,j+1]\times[i\,,i+1]$,*
- *f satisfies the average matching condition* (3)*, and*
- *f has a continuous gradient and cross-derivative, and its normal derivative vanishes at every point of the boundary of its domain (natural boundary conditions).*

Like bicubic splines, biquadratic histosplines satisfy a minimal norm property:

Theorem 1. [9] *The natural biquadratic histospline is the smooth average matching function with a cross-derivative with least RMS norm. More precisely, the natural biquadratic histospline minimizes $\int_0^m\int_0^n(\frac{\partial^2 f}{\partial x\partial y})^2 dxdy$ over all f which satisfy* Eq. (3) *in the Wiener function space $\mathrm{W}_2^{1,1}$.*

3 Fast Natural Histospline Computation and Sampling

In 1979, W. Tobler and J. Lau used a sinc-like cardinal natural biquadratic histospline basis to upsample images [10]; to the authors' knowledge, theirs is the only published reference to the use of global histosplines for image resampling. In 1993, J. Kobza and J. Mlčák published algorithms for the computation of biquadratic histospline surfaces in piecewise polynomial form for various boundary conditions and tensor grids [9]. Our method relies on B-Splines [6,11]. While local and global interpolatory splines and B-Splines have seen much use for image interpolation and smoothing—to wit the many entries in the visionbib database—we are not aware of any previous work on global or local histopolation based on B-Splines ([8] and [12] come close).

Natural biquadratic histospline resampling is fully defined by its univariate components: It is the tensor product of two univariate methods which compute quadratic histosplines [8]. In the reconstruction stage (Eq. (3)), tensor B-Spline coefficients are obtained by solving one tridiagonal linear system per image row and column. In the sampling stage (Eq. (4)), each pixel row of the resampled image is obtained from the tensor B-Spline coefficients by the "vertical" application of the linear operator corresponding to integration with respect to y—this only requires three rows of B-Spline coefficients at any given time, four if the output

row being computed overlaps two input rows—followed by the "horizontal" application, within the output row, of the analog of integration with respect to x [1]. (Note: In our implementation, reconstruction and sampling are interwoven.)

3.1 Computing Histosplines with Piecewise B-Spline Antiderivatives

The following division free [13] implementation of fast solution methods for special tridiagonal linear systems [14,15] is analogous to the fast computation of cubic splines by recursive causal and anti-causal filtering [11].

Univariate histosplines are computed as $f(x) = \sum_{j=0}^{n-1} a_j B'_j(x)$, where

$$B_0(x) = \begin{cases} x(6-x^2) & \text{on } [0,1)\,, \\ t\,(3+t\,(-3+t))\,, \ \text{where } t = x-1, & \text{on } [1,2)\,; \end{cases} \tag{6}$$

$$B_j(x) = \begin{cases} s^3, & \text{where } s = x-(j-1), \ \text{on } [j-1,j)\,, \\ t\,(3-t\,(-3+2t))\,, & \text{where } t = x-j, \ \text{on } [j,j+1)\,, \\ u\,(3+u\,(-3+u))\,, & \text{where } u = x-(j+1), \ \text{on } [j+1,j+2)\,; \end{cases} \tag{7}$$

$$B_{n-1}(x) = B_0(n-x) \text{ on } [n-2,n)\,. \tag{8}$$

Although B_j is discontinuous, B'_j is continuously differentiable. This formulation of the natural quadratic histospline basis $\{B'_j\}_{j=0}^{n-1}$, in terms of piecewise antiderivatives which vanish at every integer, leads to the accurate computation of the integrals of the basis functions over arbitrary intervals. (The discrepancy in constants of integration only needs to be taken into account when the sampling interval of integration contains an integer.) The function $f(x)$ satisfies the univariate version of the average matching condition (3) if and only if

$$\underbrace{\begin{pmatrix} 5 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 5 \end{pmatrix}}_{\mathrm{A}} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-3} \\ a_{n-2} \\ a_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & & & & \\ c_0 & 1 & & & \\ & c_1 & 1 & & \\ & & \ddots & & \\ & & & c_{n-3} & 1 \\ & & & & c_{n-2} \ 1 \end{pmatrix}}_{\mathrm{L}} \underbrace{\begin{pmatrix} d_0 & 1 & & & \\ & d_1 & 1 & & \\ & & d_2 & 1 & \\ & & & \ddots & \\ & & & & d_{n-2} \ 1 \\ & & & & d_{n-1} \end{pmatrix}}_{\mathrm{U}} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-3} \\ a_{n-2} \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-3} \\ p_{n-2} \\ p_{n-1} \end{pmatrix},$$

where

$$c_0 = \frac{1}{d_0} = \frac{1}{5}, \quad c_j = \frac{1}{d_j} = \frac{1}{4-c_{j-1}} \ (j = 1,2,\ldots,n-2), \quad c_{n-1} = \frac{1}{d_{n-1}} = \frac{1}{5-c_{n-2}}.$$

This increasing recurrence relation, which defines a continued fraction, converges exponentially [16]. In fact, c_6 is indistinguishable from the limit $2-\sqrt{3}$ in single precision, c_{14}, in double precision. Consequently, L^{-1} and U^{-1} can be hard-coded with eight constants in single precision, sixteen in double precision, and Gaussian elimination mostly involves row operations with fixed multiplier $c = 2-\sqrt{3}$: $p_j \leftarrow p_j - cp_{j-1}$ in the forward elimination stage, $p_j \leftarrow c(p_j - p_{j+1})$ in the back substitution. (For 8 and 16 bit images, truncating the c_j sequence on the basis of floating point precision is overkill: fewer c_js can be used with no ill effect.)

Forward elimination requires $n-1$ multiplications and $n-1$ subtractions, back substitution, n multiplications and $n-1$ subtractions, for a total of $2n-1$ multiplications and $2n-2$ subtractions. In order to compute the coefficients of the biquadratic histospline, one Gaussian elimination must be performed for every row and column of the image. Consequently, $4mn - m - n$ multiplications and $4mn - 2m - 2n$ subtractions—less than 8 flops per input pixel—are needed to compute tensor B-Spline coefficients for an $n \times m$ greyscale image.

3.2 Packing Partially Computed Coefficients into Small Data Types

An 8 bit color depth for input and output image pixel values is assumed in the remainder of this article.

Unlike local methods, global interpolatory and histopolating spline methods require either the use of globally defined cardinal basis functions analogous to tensor products of sinc functions [10]—for which the determination of expansion coefficients (reconstruction) is trivial but the evaluation (resampling) is costly—or the use of local bases like B-Splines [6,11] or piecewise polynomials [9], which requires the global storage of partially or fully computed expansion coefficients.

Four versions of natural biquadratic histospline upsampling are implemented in C. The versions differ in the type of the data array used to store partially computed B-Spline coefficients: double, float, uint16, or uchars. Storing the coefficients corresponding to a color channel as doubles uses eight times the memory of the color channel; using uchars, one can overwrite the image with coefficients.

We present the key features of the uchar implementation; the uint16 version, which produces enlargements essentially identical to the double and float versions, is similar. Gaussian elimination is first performed within each input image pixel column as follows: Each input pixel value, in the range $[0, 255]$, is scaled and shifted to the range $[-255, 255]$ with the affine mapping $p \mapsto 2p-255$ and stored into a float array of length m ($p \mapsto 2p-255$ can be performed in integer arithmetic: no flop required). Gaussian elimination maps $[-255, 255]$ into $(-127.5, 127.5)$. Here is a proof of this fact: If we show that $[-1, 1]$ is mapped into $(-1/2, 1/2)$, we are done. The forward elimination stage consists of

$$p_i \leftarrow p_i - c_{i-1}p_{i-1} \quad (i = 1, m-1).$$

Because the c_is are positive,

$$|p_1| \le 1 + c_0;\ |p_2| \le 1 + c_1(1 + c_0) = 1 + c_1 + c_0c_1;\ |p_3| \le 1 + c_2 + c_1c_2 + c_0c_1c_2, \text{ etc.}$$

For every i, $c_i < c = 2-\sqrt{3}$, so that $|p_i| < 1/1 - c \equiv C$. The back substitution stage consists of

$$p_{m-1} \leftarrow c_{m-1}p_{m-1}, \text{ and } p_i \leftarrow c_i(p_i - p_{i+1}) \quad (i = m-2, 0).$$

Consequently,

$$|p_{m-1}| < Cc_{m-1};\ |p_{m-2}| < c_{m-2}(C + Cc_{m-1}) = C(c_{m-2} + c_{m-2}c_{m-1}), \text{ etc.}$$

At the end of Gaussian elimination, $|p_i| < Cc/(1 - c) = 1/2$ for every i, which establishes that $\|\mathrm{A}^{-1}\|_\infty < 1/2$ [17]. This bound is asymptotically tight: the seesaw

mode comes arbitrarily close to attaining it. Now that we know that the half-computed B-Spline coefficient values are in the interval $(-127.5, 127.5)$, we can safely store them as uchars by adding 128. and casting.

The solution $\mathbf{a}$ of $\mathbf{Aa} = \mathbf{p}$ can be recovered cheaply from the solution $\tilde{\mathbf{a}}$ of $\mathbf{A}(\tilde{\mathbf{a}} - 127.5) = 2\mathbf{p} - 255$ because constant vectors are eigenvectors of $\mathbf{A}$ with eigenvalue 6, which implies that $\mathbf{a} = .5(\tilde{\mathbf{a}} - 85)$. Because the multiplication by .5 can be folded into the sampling stage at no cost—by merging .5, together with the multiplication by MN/mn, into quadrature coefficients—shifting/packing/unpacking only requires one additional flop (adding 128.). Only one such packing/unpacking is necessary, for the following reason: The pixel values of an output row depend on the B-Spline coefficients corresponding to at most four input rows. Consequently, only four rows—three with integer magnification factors—of B-Spline coefficients are needed at any time, which implies that we can afford to compute and store them in floating point. (In 3D, three or four floating point B-Spline coefficient "slabs" should be used.) B-Spline coefficient rows only need to be computed once if output pixel rows are computed from top to bottom and coefficients are computed when needed. This implies that the error introduced by packing and unpacking the coefficients in and out of uchars is minimal. The following back of the envelope estimate suggest that typically the effect of packing/unpacking on output pixel values is at most 3, as is observed in practice: Suppose that there is no other source of error besides packing/unpacking. Rounding values when packing the half-computed coefficients into uchars introduces an error $\leq .5$. Unpacking multiplies this error by .5. Row by row Gaussian elimination now puts the B-Spline coefficient error in the interval $(-1/8, 1/8)$. Assuming that the error of an output pixel comes from the error in one B-Spline coefficient, Eq. (7) implies that the largest pixel errors introduced by uchar storage are $1/8 \times (9/2)^2 = 81/32 \approx 3$ in the worst case situation $M \gg m$ and $N \gg n$. (Using a pessimistic but rigorous estimate as in §1.3 gives a bound of $441/32 \approx 14$.) Similar estimates suggest that uint16 storage introduces insignificant pixel error (no more than $81/8224 \approx .01$).

3.3 Tensor Computation of Pixel Integrals (Sampling Stage)

Because at most four contiguous quadratic B-Splines overlap any given interval of length at most 1, the linear quadrature which maps B-Spline coefficients to output pixel values is completely described by four (three for integer magnification) double arrays of length m and four (three) double arrays of length n, together with two integer arrays, of lengths $m-1$ and $n-1$, which specify relevant index ranges. Output pixel rows can be computed one at a time with about $4 \times 4 + 4 = 20$ floating point multiplications and $4 \times 3 + 3 = 16$ additions per output pixel value ($3 \times 3 = 9$ multiplications and $3 \times 2 + 2 = 8$ additions if both magnification factors are integers), the same as local bicubic resampling. (Taking non-overlapping output pixels into account lowers the flop count.)

3.4 Overall Computation Cost

About $8mn + 36MN$ flops per color channel are needed to enlarge an image from dimensions $n \times m$ to $N \times M$ ($8mn + 17MN$ flops if m divides M and n divides N). Our

GIMP natural biquadratic histopolation plug-ins upsample large images more rapidly than the built-in bicubic resampler but more slowly than the built-in bilinear one.

4 Quantitative Comparison with Other Linear Methods

Twenty linear resampling methods are compared to the double, float, uint16 and uchar versions of natural biquadratic histopolation: box filtering, natural and not-a-knot bicubic spline interpolation, and seventeen ImageMagick filters used with default settings. The test suite is set up so that errors do not originate from image size convention mismatch (the "center" convention $N \times M$ vs. the "corner" convention $(N-1) \times (M-1)$): ImageMagick uses the same convention as this article's AM methods (this is undocumented: see the resize.c source code); our Scilab/SIVP implementations of cubic spline interpolation use the $N \times M$ image size convention as well. Although the "center" image size convention is not the most commonly used for interpolatory resampling, this levels the field.

4.1 Test Setup

Ten copyfree digital images—photographs and scans of small objects (J.-F. Avon), astronauts and spacecraft (NASA), a woodcut print of a wave (K. Hokusai), a chapel (M. Ryckaert), a katydid (wikipedia user wadems), a seated man (S. Prokudin-Gorskii), a vervet in a tree (W. Welles), as well as close ups of a baby (M. Gong) and a man (A. Adams)—are cropped to 1680×1680. The crops are then downsampled with box filtering to 840×840, 560×560, 420×420, 336×336, 280×280, 240×240 and 210×210. Downsampling by an integer factor with box filtering mimics the image capture process; more importantly, it does not introduce error (other than rounding). For this reason, the downsampled versions of the cropped originals are treated as if error-free. For the integer magnification tests, they are enlarged back to 1680×1680. For the rational magnification tests, they are enlarged to the next larger size; for example, tests with magnification factor 3/2 are performed by enlarging 560×560 images to 840×840. Error measures are computed by comparing the re-enlargements to the cropped originals (integer magnification) or their downsampled versions (rational magnification). Four carefully implemented error metrics are used: Root Mean Squared Error (RMSE), Average Absolute Error (AAE), Maximum Absolute Error (MAE), and Mean Structural SIMililarity index (MSSIM) [18], analogous to a correlation in that larger MSSIMs correspond to smaller errors. The seventy integer (for each method) magnification results (one per test image and integer magnification) are amalgamated as follows: the RMSEs by taking the square root of the mean of their squares, the AAEs, MAEs and MSSIMs by plain averaging; likewise for the sixty rational magnification results. Making exceptions for box filtering, nearest neighbor interpolation and the uchar version of natural biquadratic histopolation, we omit results for methods which performed more poorly than bilinear interpolation in at least one of the

Table 1. Test results: linear upsampling methods ranked by increasing RMSE

Test results for the rational magnification factors $\frac{8}{7}$, $\frac{7}{6}$, $\frac{6}{5}$, $\frac{5}{4}$, $\frac{4}{3}$ and $\frac{3}{2}$

Resampling method	RMSE	AAE	MAE	MSSIM
Natural biquadratic histospline (uint16)	4.9453524	2.3502549	68.7	.9679117
Natural biquadratic histospline (float)	4.9453652	2.3502540	68.7	.9679115
Natural biquadratic histospline (double)	4.9453653	2.3502542	68.7	.9679115
ImageMagick Hamming (windowed sinc)	4.9654332	2.4536258	69.0	.9669423
Scilab-SIVP natural bicubic spline	4.9746911	2.3496956	69.7	.9671314
ImageMagick Lanczos (3-lobes Lanczos)	4.9768380	2.4540594	69.0	.9670086
Scilab-SIVP not-a-knot bicubic spline	4.9772149	2.3514317	69.7	.9671094
ImageMagick Kaiser (windowed sinc)	4.9794970	2.4483523	69.4	.9669581
ImageMagick Hanning (windowed sinc)	4.9799794	2.4582932	69.2	.9668586
Natural biquadratic histospline (uchar)	4.9830553	2.4759742	68.9	.9652749
ImageMagick Blackman (windowed sinc)	5.0076199	2.4463746	70.0	.9668665
ImageMagick Welsh (windowed sinc)	5.0099166	2.5068305	69.3	.9659990
ImageMagick Parzen (windowed sinc)	5.0482158	2.4516398	70.7	.9666239
ImageMagick Catrom (Catmull-Rom)	5.2415191	2.5088389	73.4	.9650689
ImageMagick Lagrangian (bicubic)	5.3354821	2.5643617	74.2	.9636494
ImageMagick Mitchell (Mitc.-Netravali)	5.8867701	2.8302989	78.7	.9579545
Box filtering	6.1401973	2.7962816	82.1	.9575975
ImageMagick Hermite (w/ $\nabla f(x_j, y_i) = 0$)	6.1427426	2.8683378	82.0	.9572294
ImageMagick Triangle (bilinear)	6.2280344	2.9553329	83.3	.9547236
ImageMagick Point (nearest neighbor)	8.2601463	3.5089177	106.6	.9413052

Test results for the integer magnification factors 2, 3, 4, 5, 6, 7 and 8

Resampling method	RMSE	AAE	MAE	MSSIM
Natural biquadratic histospline (uint16)	9.7425786	4.4851986	139.4	.8450685
Natural biquadratic histospline (double)	9.7425817	4.4851985	139.4	.8450685
Natural biquadratic histospline (float)	9.7425817	4.4851987	139.4	.8450685
ImageMagick Hamming (windowed sinc)	10.026813	4.6403736	140.6	.8393992
ImageMagick Welsh (windowed sinc)	10.036087	4.6717838	140.6	.8379616
ImageMagick Lanczos (3-lobes Lanczos)	10.042201	4.6440940	140.8	.8394069
ImageMagick Hanning (windowed sinc)	10.051135	4.6523516	140.6	.8393168
ImageMagick Kaiser (windowed sinc)	10.055531	4.6466714	140.9	.8395413
Scilab-SIVP natural bicubic spline	10.078684	4.6170044	141.0	.8396235
Scilab-SIVP not-a-knot bicubic spline	10.081273	4.6188794	141.0	.8395692
ImageMagick Blackman (windowed sinc)	10.088110	4.6523029	141.1	.8396107
ImageMagick Parzen (windowed sinc)	10.123885	4.6610338	141.6	.8395226
ImageMagick Catrom (Catmull-Rom)	10.274565	4.7180942	143.1	.8380914
ImageMagick Lagrangian (bicubic)	10.388237	4.7850457	143.9	.8354883
ImageMagick Mitchell (Mitc.-Netravali)	10.826885	5.0324787	146.6	.8298020
ImageMagick Hermite (w/ $\nabla f(x_j, y_i) = 0$)	10.851653	4.9917407	148.8	.8305381
ImageMagick Triangle (bilinear)	11.020940	5.1227437	149.5	.8273934
Natural biquadratic histospline (uchar)	12.159461	5.1615913	139.4	.8429333
Box filtering = ImageMagick Point	12.172721	5.3349523	165.4	.8023312

two groups of tests: the ImageMagick methods Bessel (windowed jinc), Gaussian (Gaussian blur), and Quadratic and Cubic (polynomial approximations of Gaussian blur).

4.2 Test Results

As seen in Table 1, the double, single and uint16 versions of natural biquadratic histopolation best the other methods with respect to every error metric, with a single exception, natural bicubic spline interpolation, which gets a lower AAE in the (small) rational magnification tests. This suggests that natural biquadratic spline histopolation may be the most accurate reconstructor.

Here is a brief discussion of subjective image quality. Enlargements computed with natural biquadratic histopolation are unquestionably the sharpest: Small details really stand out. However, they show a lot of haloing, probably the most of all tested methods. Aliasing is also noticeable, although less so than with some of the other methods. This suggests that a box filtered version of natural biquadratic histopolation may yield more visually pleasing enlargements.

Acknowledgements. Research funded by Canada NSERC Discovery and USRA and Ontario CFI New Opportunity grants. We thank Michael Herman, Steven A. Ruzinsky, Michael Unser and Jean-François Avon for useful discussions.

References

1. Dodgson, N.A.: Image resampling. Technical Report UCAM–CL–TR–261, University of Cambridge Computer Laboratory, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK (1992)
2. Kuhnert, K.D.: Sensor modeling as basis of subpixel image processing. In: Duverney, J.F. (ed.) Proceedings SPIE Image Processing III, Paris France, vol. 1135, pp. 104–111 (1989)
3. Park, S.C., Park, M.K., Kang, M.G.: Super-resolution image reconstruction: a technical overview. Signal Processing Magazine, IEEE 20, 21–36 (2003)
4. Price, J., Hayes, I.M.H.: Sensor optimal image interpolation. In: Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, vol. 2, pp. 1262–1266 (1999)
5. Zukowski, J.: Java AWT reference. O'Reilly, Sebastopol (1997)
6. de Boor, C.: A practical guide to splines. Applied Mathematical Sciences, vol. 27. Springer, New York (1978)
7. Gentle, J.E.: Elements of Computational Statistics. In: Statistics and Computing, 1st edn., Springer, Heidelberg (2002)
8. Aràndiga, F., Donat, R., Mulet, P.: Adaptive interpolation of images. Signal Process 83, 459–464 (2003)
9. Kobza, J., Mlčák, J.: Biquadratic spline smoothing mean values. Acta Univ. Palack. Olomuc. Fac. Rerum Natur. Math. 33, 339–356 (1994)
10. Tobler, W., Lau, J.: Interpolation of images via histosplines. CGIP 9, 77–81 (1979)
11. Unser, M.: Splines: a perfect fit for signal/image processing. IEEE Signal Process. Magazine 16(6), 22–38 (1999)

12. Heckbert, P.: Filtering by repeated integration. Computer Graphics 20(4), 315–321 (1986)
13. Anderson, N.: The L(D**(-1),U) decomposition. NA Digest 93(13) (1993)
14. Malcolm, M.A., Palmer, J.: A fast method for solving a class of tridiagonal linear systems. Commun. ACM 17(1), 14–17 (1974)
15. Boisvert, R.F.: Algorithms for special tridiagonal systems. SIAM J. Sci. Stat. Comput. 12(2), 423–442 (1991)
16. Hafner, J.L.: Explicit and asymptotic formulas for LDMT factorization of banded Toeplitz matrices. Linear Algebra Appl. 222, 97–126 (1995)
17. Golub, G., Van Loan, C.: Matrix Computations, 2nd edn. The Johns Hopkins University Press, Baltimore (1989)
18. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE T. Image Proces. 13(4), 600–612 (2004)