

Zvi Biener

## Fast Algorithms for Image Magnification and Demagnification Abstract

The project presents analysis and implementation of several image magnification (interpolation) and demagnification (decimation) techniques for non-integer magnification/demagnification factors. Fast algorithms for image magnification and demagnification which are based on fast sinc interpolation algorithm [1,2,3] are described. These algorithms exhibit higher performance than the traditional methods. Implementation and performance issues are discussed. Experimental evidence to support the claims are provided.

### 1. Block diagram of fast sinc interpolation algorithm for odd N

The fast sinc-interpolation algorithm for odd N (N is the number of coefficients used in fast Fourier transform) is presented in fig. 1:

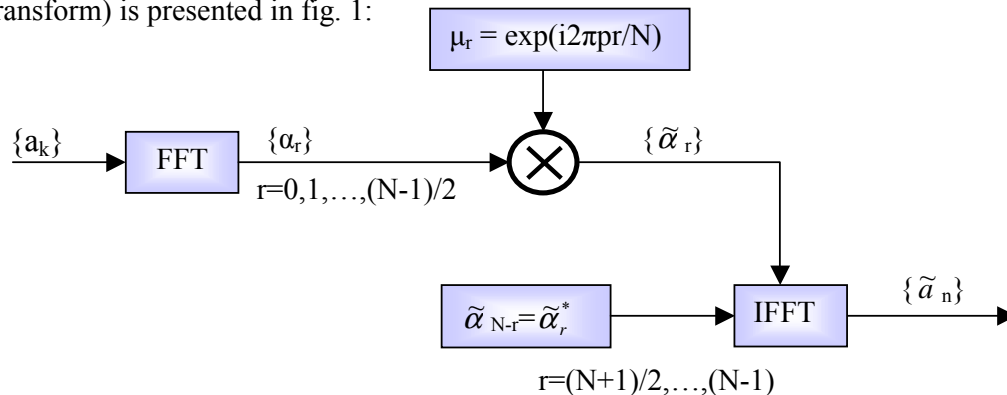


Fig. 1: Block diagram of the fast sinc-interpolation algorithm.

This algorithm is described in details on ref.[1,2,3].

### 2. Block diagram of demagnification system based on sliding window fast sinc interpolation in frequency domain.

The principle of sinc-interpolation in sliding window is illustrated in Fig.2 for the filter size of  $N=17$ . The black points at the bottom are 17 pixels from the original image (before decimation). Each pixel (of the 17 pixels) value is multiplied with the sinc value according to its position, then all these resulted values are summed up and the result is the value of the target image pixel, which is standing exactly under the sinc peak value.

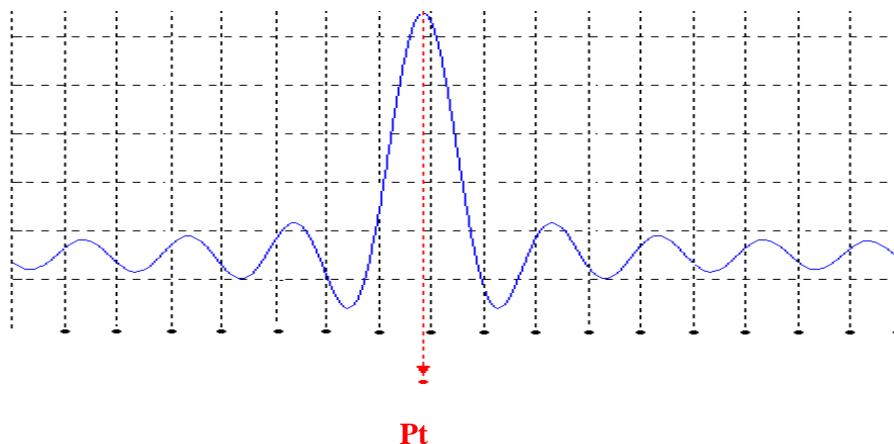


Fig. 2: Example of  $N=17$  SINC window filter.

The fast sinc decimation algorithm (see Fig. 4) uses the fast sinc-interpolation algorithm (see Fig. 1) for mapping pixels of the original image into pixels of the target (decimated) image and it does it with a sliding window. In case of very small demagnification factor there will not be unnecessary interpolation computation far from the target pixel position because the window will always slide until the target pixel position gets to the center of the window and only then the interpolation calculation are made. The sliding window size (N) is an odd number.

From each window, which covers N values in the original image in DFT domain, one value is created in the image domain (using the center coefficient of the IDFT transform:  $n=(N-1)/2$ ).

$p$  parameter from Fig. 4 is defined as the following relation (illustrated in Fig. 3):

$$p_1 = x_1/x_0$$

$$p_2 = x_2/x_0$$

$$p_3 = x_3/x_0$$

$x_0$  is the distance between each 2 adjacent pixels (big dots) in the original image.

The little dots are the pixels of the decimated image.

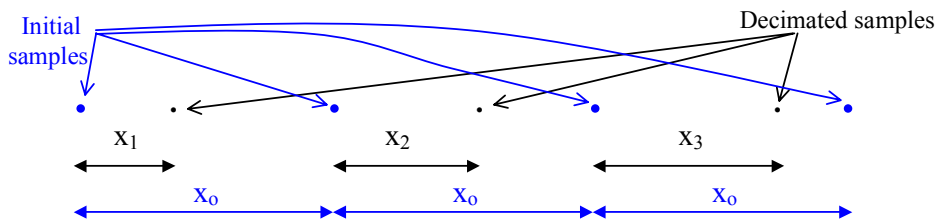


Fig. 3: Signal decimation in sliding window

The decimation filter can be implemented in DFT domain as it is shown in Fig. 4.

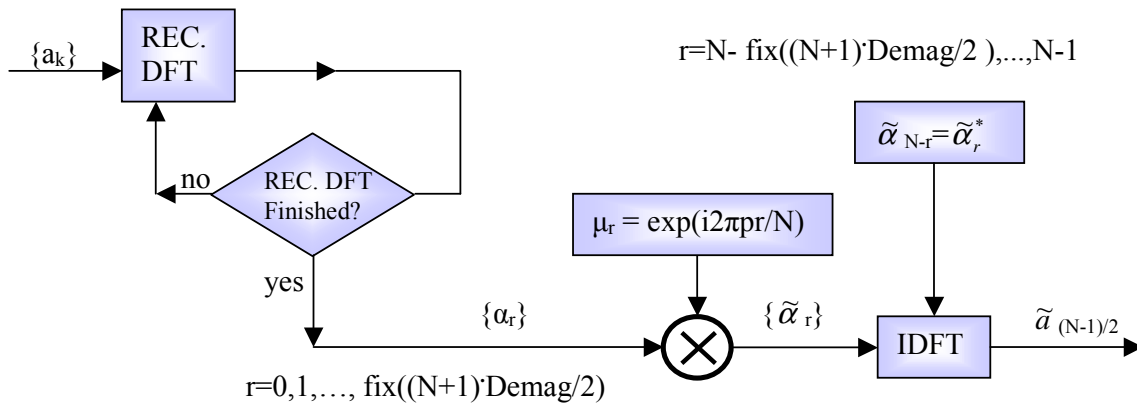


Fig. 4: Signal decimation algorithm using fast sinc-interpolation  
**Demag** is the demagnification factor ( $0 < \text{Demag} \leq 1$ ).

Implementation of the magnification on the basis of fast sinc interpolation algorithm (see Fig. 1) is similar to the demagnification system of fig. 4 except for the absence of low pass filtering. In each of the sliding window position all the interpolated values between the original image pixel at the window center and the next one should be calculated.

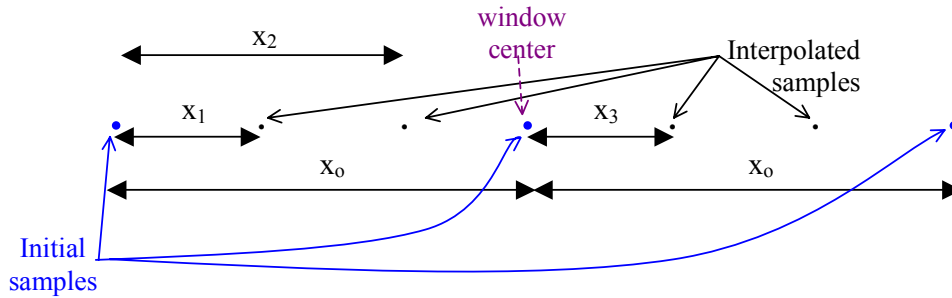


Fig. 5: Signal magnification in sliding window

Magnification and demagnification can also be implemented in signal domain. It can be done by multiplying the sinc weights with the original image pixel of the current sliding window and thus creating the interpolated pixel at the desired position. In the signal domain image interpolation algorithms run faster as shown in the calculations complexity sec. in [5]

### 3. Reducing program running time

Computing values of cosine or sine of a certain angle is equivalent to several multiplication and addition operations. To save program running time for an image with size of  $M1 \times M2$  it is possible to calculate the values of all the sine and cosine coefficients for only the first row in the image because the positions of the pixels in the first row are exactly the same as in all the other rows for separable row-column interpolation. The computed values are stored in a buffer (Look up table) to be used for all the image rows (see Fig. 6).

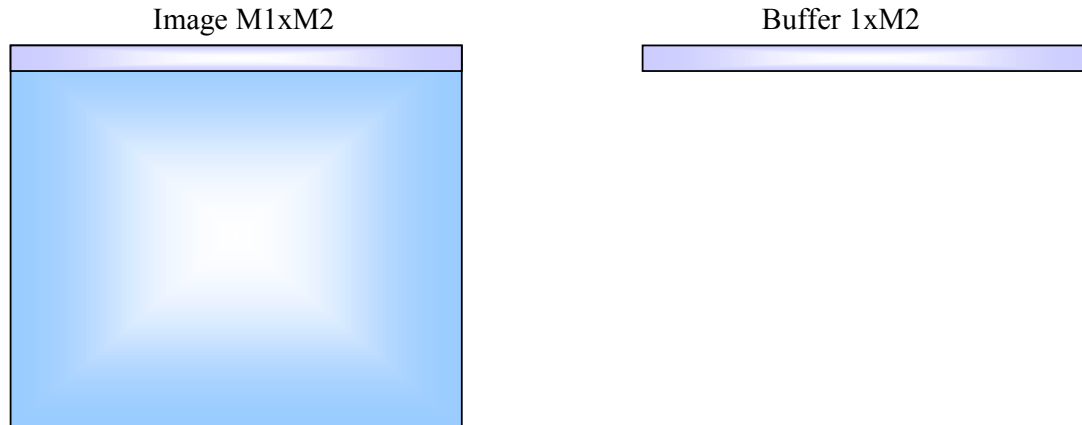


Fig. 6: The use of a buffer which holds sine/cosine calculated values of the first image row is used for the entire image to reduce programming running time.

### 4. Summary of performance results

An experiment was made to evaluate the performance of the sinc window. In this experiment an image of Lenna with size of  $228 \times 228$  (fig. 19 in [5]) was used. The image was decimated and then magnified back to  $228 \times 228$  by interpolation in different methods. The decimation was done by the frequency domain decimation algorithm which uses fast sinc-interpolation (see sec. 2).

Two values were calculated based on the difference image of each method:

$$1. \text{ Square error: } e^2 = \sum_{n=0}^{pxl \text{ sum}-1} (|p_t[n]| - |p_s[n]|)^2$$

**pxl sum** is the number of pixels in the target image.  $\{p_t\}$  are target image values.  $\{p_s\}$  are source image values.

$$2. \text{ Standard deviation: } \sigma = (e^2 / (pxl \text{ sum}))^{1/2}$$

The next table (table 1) summarizes the performance results for the 5 magnification methods for magnification factors of 7.6, 3.8.

Interpolation method	Standard deviation for magnification factor of <b>7.6</b>	Standard deviation for magnification factor of <b>3.8</b>
nearest neighbour	30.61	23.63
linear	27.17	21.03
bicubic	26.63	20.25
sinc-interpolation window	26.13	19.90
Hanning window interpolation [4,5]	26.18	19.88

Table1: Performance results for the 5 magnification methods for magnification factor of 7.6, 3.8.

From comparison of the last 5 magnification methods fast sinc-interpolation method and Hanning window give the best results. The difference between Hanning method results and sinc method results is negligible.

## 5. References

- [1] L. P. Yaroslavsky, Efficient algorithm for discrete sinc-interpolation, Applied Optics, VOL. 36, No.2, 10 January 1997, p. 460-463.
- [2] L. P. Yaroslavsky, Fast sinc-interpolation and its applications in signal and image processing' 1s &T/SPIE's 14<sup>th</sup> annual symposium, Electronic imaging 2002, Proc. SPIE, v.4667.
- [3] L. P. Yaroslavsky, Lecture notes, [www.eng.tau.ac.il/~yaro](http://www.eng.tau.ac.il/~yaro).
- [4] Philippe Thevenaz, Member, IEEE, Thierry Blu, Member, IEEE, AND Michael Unser, Fellow, IEEE, Interpolation Revisited, IEEE Transactions On Medical Imaging, VOL.19, No.7, July 2000.
- [5] Zvi Biener, project submitted toward the degree of master of science in engineering science, Tel-Aviv un. , June 2002.