NON-UNIFORM KERNEL SAMPLING IN AUDIO SIGNAL RESAMPLER

Grzegorz Kraszewski

Białystok Technical University, Electrical Engineering Faculty,
ul. Wiejska 45D, 15-351 Białystok, Poland, e-mail: krashan@teleinfo.pb.bialystok.pl

Abstract: *Classic interpolators (resamplers) of band-limited signals are based on windowed sinc() function kernels. To speed computations up, kernel is not calculated in realtime, but sampled and stored in a table. Linear interpolation between kernel samples is used to lower noise caused by discretization of the continuous kernel function. In spite of this, the table is lenghty for high quality interpolators. This paper describes how the table size can be significantly lowered at a cost of slightly increased computation complexity and without quality loss. The algorithm described is specifically designed for SIMD processors, which high computational power is limited by memory bandwidth.*

1 Introduction

Resampling of a bandlimited signal is a common task in signal processing, especially in audio signals processing. The general discussion of resampling systems is given in [1] and [2]. The input to a resampler is a discrete signal $x[n]$, obtained by sampling original continuous $x(t)$ with period T_{in}. The first step of resampling is reconstruction of original $x(t)$. Then, if sampling rate is to be decreased, an antialiasing lowpass filter must be applied, so reconstructed signal bandwidth is limited to $f_{out}/2$, where f_{out} is the output sampling frequency. The last stage of resampling is to sample reconstructed and if neccesary filtered $x(t)$ at the new sampling rate. There are different approaches to $x(t)$ reconstruction problem. The most straighforward approach is given in [3], where input signal is convolved with continuous *sinc*() kernel. While theoretically ideal, it is impossible to implement, because of infinite *sinc*() function support and computation cost of *sinc*(). The first problem may be solved by windowing the kernel, usually with Kaiser [3], or Chebyshev-Dolph [4], [5] window. The second is solved by sampling windowed kernel and store it in a look-up table. In fact this solution is a particular case of GASRC [2], where upsampling rate is equal to kernel oversampling (usually power of two [3], [4]), digital interpolator filter is approximated brickwall one (hence windowed *sinc*() impulse response), analog interpolator is sample and hold or linear one.

It has been shown in [6], that windowed *sinc*() interpolator cannot be considered optimal for every application, as it can't for example resample constant signal precisely (which may be critical in image processing). It is not a problem in audio processing however as passband starts from 20 Hz here. Windowed *sinc*() interpolator has also an advantage of easy control of distortions in the frequency domain, as three sources of distortions: kernel windowing, kernel interpolation and aliasing may be controlled independently [4]. Another advantage of the solution is the possibility to change resampling ratio in real time and wide range (for example multioctave pitch sliding effects), which is not possible in the algorithm proposed in [1]. Windowed *sinc*() resampling is also easy to paralellize for execution on SIMD processors. It has a drawback however, which is usage of large lookup table containing oversampled kernel. A computing power of current SIMD processors can be limited by memory bandwidth, so large lookup tables should be avoided. For example PowerPC G4 AltiVec unit clocked at 1.0 GHz, is able to perform up to 10^9 MAC (multiply and accumulate) instructions per second taking 8 floating point arguments each [7]. It gives an 32 GB/s stream of data, which is beyond capabilities of memory systems in consumer electronics/PC-s. Lowering the size of lookup tables even at a cost of increased number of calculations, yields increased performance of an algorithm executed by SIMD processor [9].

2 Kernel sampling

The most precise way to obtain a value of a continuous reconstruction kernel at any point x is to calculate it in-place. It involves one division and calculation of $sin(x)$ function, which is usually based on Taylor series. Distortions caused by kernel sampling are eliminated this way, but calculations take time. Things get worse if the *sinc*() function is windowed. The Kaiser window requires calculation of the modified Bessel function, which is again developed in power series. The Chebyshev-Dolph window is given in the frequency domain and usually transformed with DFT, so no continuous time domain representation exists even.

The first approach to kernel interpolation was nearest-neighbour interpolation between the look-up table entries. Some results are cited in [1] and given in [4]. To suppress sidelobes level below -80 dB, the kernel must be oversampled 4096 times, so depending on requirements on filter transition band it may result in 32 768 [4] – 140 000 [1] samples. Linear interpolation between samples is an obvious step forward, it improves distortions level by factor of 2 (in dB scale) [4], so oversampling by 128 is

enough to achieve -100 dB attenuation of distortions. Higher level interpolation is investigated in [8] with the difference that not filter kernel is interpolated, but oversampled input sequence. The goal of presented algorithm is to increase interpolation precision while staying on linear (1-st order) interpolation level. It can be done with non-uniform kernel sampling. Looking at sampled windowed *sinc*() (abbreviated later in this paper as *wsinc*()) kernel it is intuitive that sample points should be placed more dense in some areas, while in other places the kernel may be approximated by linear function in wider range. Let's constrain samples x position to be aligned to $\pi/2^N$ grid, N = 10 on the following diagrams. Sample positioning starts from 0, of course $wsinc(0) = 1.0$. Then x is increased in $\pi/2^N$ steps while following inequality is true:

$$\varepsilon = \sum_{i=k+1}^{n-1} \left| wsinc\left(\frac{i\pi}{2^N}\right) - g(i,k,n) \right| < \varepsilon_0 \qquad (1.1)$$

where $g(i,k,n)$ is a linear interpolator between grid point of index k (the last placed point) and current grid point of index n, given by a formula:

$$g(i,k,n) = wsinc\left(\frac{k\pi}{2^N}\right) + \\ + \frac{(i-k)}{(n-k)}\left(wsinc\left(\frac{n\pi}{2^N}\right) - wsinc\left(\frac{k\pi}{2^N}\right)\right) \qquad (1.2)$$

The interpolation and formulas (1.1) and (1.2) are illustrated graphically on fig. 1.

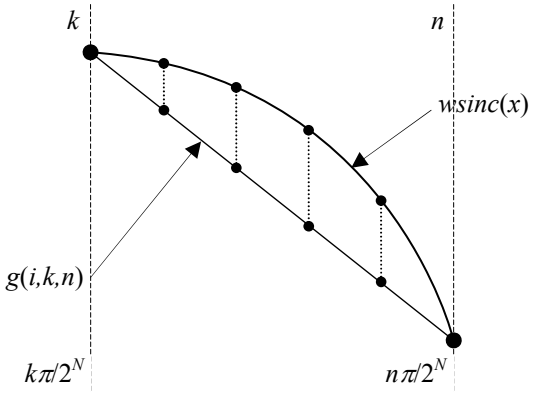

Fig. 1. Interpolation and interpolation error of the kernel.

The formula (1.1) is just the sum of lengths of dotted vertical lines on the figure 1, representing errors of linear interpolation of *wsinc*() kernel at grid points. It can be interpreted as a rough approximation of interpolation error integral:

$$\int_{\frac{k\pi}{2^N}}^{\frac{n\pi}{2^N}} \left| wsinc(x) - g(x,k,n) \right| dx \qquad (1.3)$$

where $g(x,k,n)$ is a continuous version of linear interpolator between two kernel samples.

The end point of current interpolation interval is moved right (n is increased by one) until ε exceeds ε_0. This way the sum of interpolation errors made between two consecutive kernel samples is limited to ε_0. Two diagrams below (fig. 2 and 3) shows the same windowed sinc() kernel of <0, 16π> range windowed by the Kaiser window (β = 11.0, which gives windowing sidelobes at -112 dB level). Threshold ε_0 value has been choosen as 0.0159, to get exactly 128 samples, the same number as in regulary, 8 times oversampled kernel.

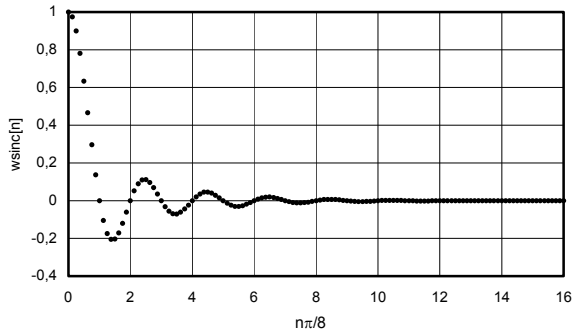

Fig.2. The kernel sampled regularly with oversampling rate of 8 (128 samples).

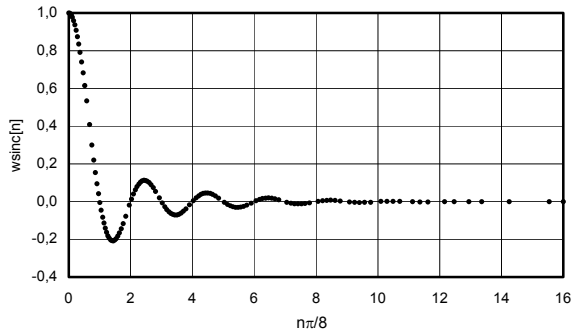

Fig. 3 The kernel sampled irregularly on π/1024 grid, (128 samples).

The figure 4 compares interpolation errors of both interpolators. The interpolation error of the regularly sampled kernel is shown in gray, its maximum value is $6.4156 \cdot 10^{-3}$. The interpolation error of the irregularly sampled kernel is shown in black, its maximum value is $6.3445 \cdot 10^{-4}$, so it is about 10 times lower. The most important thing for audio processing however, is how it influences the kernel frequency response.

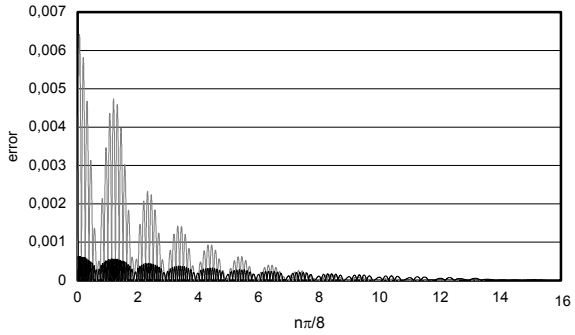

Fig.4. Comparision of interpolation errors of uniformly and non-uniformly sampled kernel (128 samples).

4 Frequency properties

Interpolation of regularly sampled kernel produces a pair of sidelobes located around the doubled oversampling frequency [4] (16 f_s in the example case), see the figure 5. These sidelobes are mirrored and repeated in both directions around every multiply of the oversampling freqency, so one of them is always reproduced in the baseband, creating aliasing distortions. The level of sidelobes expressed in the logarithmic scale depends linearly on the exponent N defining oversampling rate.

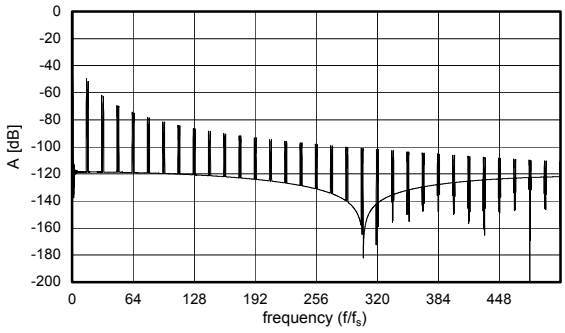

Fig. 5. Frequency response of uniformly sampled windowed sinc() kernel (128 samples).

If the kernel is sampled non-uniformly with the proposed algorithm, energy of sidelobes is spreaded on the whole frequency range between 0 and $2^{N-1} f_s$. Of course part of it still creates aliasing distortions in the passband, but most of the energy will be filtered out. In the presented example aliasing distortions caused by reconstruction kernel sampling are reduced by 18 dB (figure 6).

The more general comparision between uniform sampling and the proposed algorithm is shown on the figure 7. Four cases were investigated, the first two are kernels windowed by a Kaiser window of β = 11.0, which ensures -112 dB level of sidelobes (this level is marked on the diagram with a dashed line). Then, to remove the influence of a Kaiser window sidelobes, another Kaiser window of

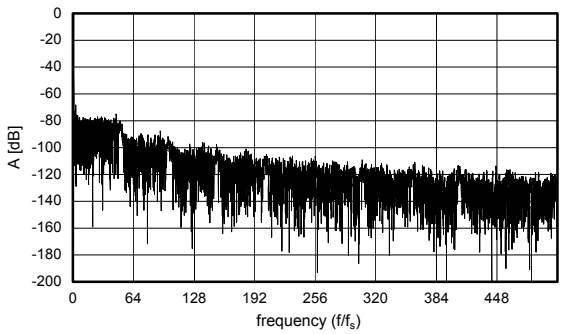

Fig. 6. Frequency response of non-uniformly sampled windowed sinc() kernel (128 samples).

β=15.0, has been used (sidelobes at -143,7 dB level). When the uniform sampling is in use, level of sampling sidelobes can be considered independent of level of window sidelobes, because they are separated in the frequency domain. This is not the case for kernel sampled non-uniformly however. The sampling sidelobes energy is scattered over all the frequency range, so part of it adds to window sidelobes. It is easy visible on the fig. 7, classic sampling S/N ratio does not depend on Kaiser β parameter (and consequently on windowing sidelobes level), while non-uniform sampling S/N ratio is decreased by 2 dB. In practical applications it must be compensated by choosing window function with lower sidelobes (at a cost of transition band width).

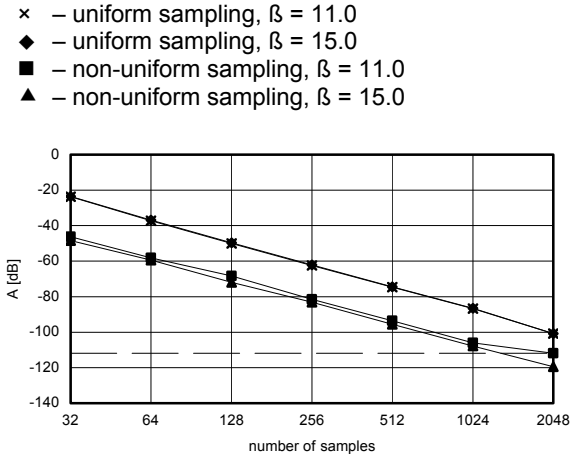

Fig. 7. Comparision of S/N ratio for uniformly and non-uniformly sampled kernels with different number of samples.

The gain of using the proposed algorithm instead of uniform kernel sampling is independent of number of samples and is equal to 18 dB. Alternatively, when S/N ratio is specified as design parameter, the same S/N may be achieved with number of saples reduced by a factor of 2.5.

5. Computational costs

Althought both interpolation algorithms, the uniform one and the proposed non-uniform one, use linear interpolation, the later is a bit more complicated, because of interpolator x interval is not fixed, but is a multiply of the base grid ($\pi/2^N$). It implies two tables must be used, one containing values of kernel at sample points, and the second one containing x coordinates of sample points. Alternatively the second table can contain integer grid indexes instead of floating point values of continuous x.

Another difficulty in the proposed algorithm is finding a pair of surrounding sample points for an arbitrary x. In the case of resampling algoritm proposed in [3], x is increased step by step while computing a single output sample. Then x lookup table may be searched starting from the latest x position instead of the beginning. After the two points surrounding current x position are found, linear interpolation between them is performed exactly the same way (and with the same number of operations), as with uniform kernel sampling.

6. Design example

A real life resampler design is presented to verify the algorithm described in the paper. The design goal for a resampler is to be "transparent" for 16-bit linear PCM data. To achieve this, level of distortions introduced by the resampler must be lower than level of the quantization noise. The latter is -97 dB for 16-bit sample resolution. Then -100 dB S/N ratio is set as the design goal for the resampler. Such a ratio requires (see the figure 7) 2048 uniform samples. If non-uniform sampling is used, 768 samples is enough. Number of samples can be further decreased by increasing the Kaiser window β parameter, but it is assumed, passband of the resampler should end at least at 0.45 f_s, so β should be keept as low as possible. For β = 11.5 (continuous kernel sidelobes at -115 dB), and choosing $6.45 \cdot 10^{-5}$ as ε_0, we get exactly 768 samples and S/N ratio -100.08 dB. Diagrams 8. and 9. show overall

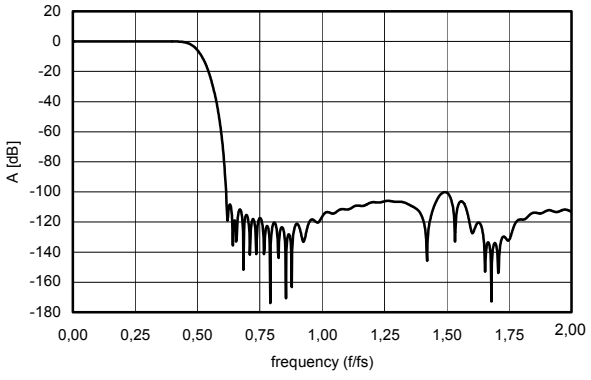

Fig. 8. The passband of example resampler design.

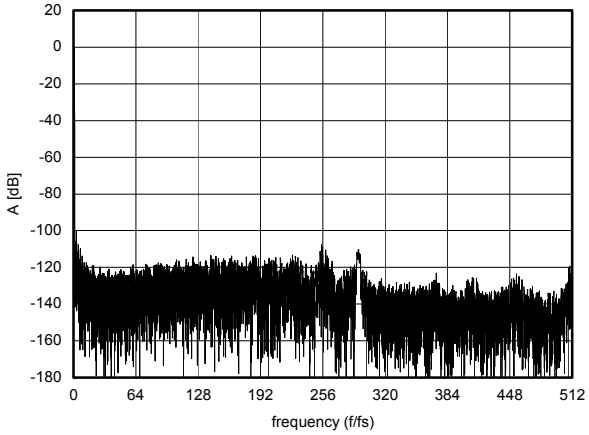

Fig. 9. The whole frequency characteristic of the example resampler design.

frequency response of the resampler and the passband in detail.

7 Conclusion and further work

The paper describes an effective method to reduce the size of the lookup table in an audio signal resampler based on continuous signal reconstruction with windowed *sinc*() function. The size reduction factor of 2.5 is achieved without quality loss. As large tables are the usual bottlenecks for algorithms executed on SIMD processors, the algorithm described can give significant speedup of resampling performed on SIMD unit.

The next step in the work will be implementing described algorithm and verifying experimentally claimed quality and properties. There is also place for further algorithm improvement. Current algorithm version just spreads the sidelobes energy on the whole frequency range including the band of resampled audio signal. It may be possible to actively shape kernel sidelobes to concentrate their energy out of audio band, then it will be filtered out by analog antialiasing filter placed at the output of an audio system. If succesfull it can give additional gain on lookup table size.

8 References

[1] Russel A. J., Beckmann P. E., *Efficient arbitrary sampling rate conversion with recursive calculation of coefficients*, IEEE Trans. on Signal Processing, vol. 50, pp. 854-865, April 2002.

[2] Evangelista G., *Design of Digital Systems for Arbitrary Sampling Rate Conversion*, EURASIP J. Signal Processing, vol. 83, no. 2, pp. 377-387, February 2003.

[3] Smith J. O., Gosset P., *Digital Audio Resampling Home Page*, http://www.ccrma-stanford.edu/~jos/

resample, 2004.
[4] Soras de L., *The Quest For the Perfect Resampler*, http://ldesoras.free.fr/doc/articles/resampler.pdf, 2003.
[5] Lynch P., *The Dolph-Chebyshev Window: A Simple Optimal Filter*, Monthly Weather Review vol. 125, 1997.
[6] Meijering E., *A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing,* Proceedings of the IEEE, Vol.90, 3.March 2002.
[7] ——, *MPC7450 RISC Microprocessor Family Reference Manual*, Freescale Semiconductor Inc., 2005.
[8] Niemitalo O., *Polynomial Interpolators for High-Quality Resampling of Oversampled Audio*, 2001.
[9] Talla D., John L. K., Burger D., *Bottlenecks in Multimedia Processing with SIMD style Extensions and Architectural Enhancements,* IEEE Transactions on Computers, vol. 52, no. 8, pp. 1015-1031, August 2003.
[10] Diefendorff K., Dubey P. K., Hochsprung R., Scales H., *AltiVec Extension to PowerPC Accelerates Media Processing*, IEEE MICRO, Mar-Apr 2000, pp. 85-95.