Downsampling and applying a lowpass filter to digital audio

Asked 16 years, 1 month ago Modified 6 years, 9 months ago

Viewed 29k times

13

I've got a 44Khz audio stream from a CD, represented as an array of 16 bit PCM samples. I'd like to cut it down to an 11KHz stream. How do I do that? From my days of engineering class many years ago, I know that the stream won't be able to describe anything over 5500Hz accurately anymore, so I assume I want to cut everything above that out too. Any ideas? Thanks.

Update: There is some code on this page that converts from 48KHz to 8KHz using a simple algorithm and a coefficient array that looks like { 1, 4, 12, 12, 4, 1 }. I think that is what I need, but I need it for a factor of 4x rather than 6x. Any idea how those constants are calculated? Also, I end up converting the 16 byte samples to floats anyway, so I can do the downsampling with floats rather than shorts, if that helps the quality at all.

audio signal-processing pcm downsampling

edited Mar 26, 2010 at 17:18

Jon Seigel

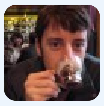
12.4k ● 8 ● 60 ● 93

Share

Improve this question

Follow

asked Oct 26, 2008 at 18:12



twk

17.3k ● 24 ● 77 ● 98

The code is a weighted average over a window. It is probably calculated from a inverse DFT of a filter function on frequency domain. You can use a 4 element array and experiment with the values, something like [0.1, 0.4, 0.4, 0.1] , until it sounds good. – artificialidiot Oct 26, 2008 at 19:02

10 Answers

Sorted by: Highest score (default)

10

Read on FIR and IIR filters. These are the filters that use a coefficent array.

If you do a google search on "FIR or IIR filter designer" you will find lots of software and online-applets that does the hard job (getting the coefficients) for you.

EDIT:

This page here (http://www-users.cs.york.ac.uk/~fisher/mkfilter/) lets you enter the parameters of your filter and will spit out ready to use C-Code...

edited Oct 26, 2008 at 18:52

Share Improve this answer

Follow

answered Oct 26, 2008 at 18:46

Nils Pipenbrinck

86.2k ● 33 ● 155 ● 223

Could you please briefly explain or link to the "hard job"? – some_id Aug 4, 2013 at 7:26

5

You're right in that you need apply lowpass filtering on your signal. Any signal over 5500 Hz will be present in your downsampled signal but 'aliased' as another frequency so you'll have to remove those before downsampling.

It's a good idea to do the filtering with floats. There are fixed point filter algorithms too but those generally have quality tradeoffs to work. If you've got floats then use them!

Using DFT's for filtering is generally overkill and it makes things more complicated because dft's are not a contiuous process but work on buffers.

Digital filters generally come in two tastes. FIR and IIR. The're generally the same idea but IIF filters use feedback loops to achieve a steeper response with far less coefficients. This might be a good idea for downsampling because you need a very steep filter slope there.

Downsampling is sort of a special case. Because you're going to throw away 3 out of 4 samples there's no need to calculate them. There is a special class of filters for this called polyphase filters.

Try googling for polyphase IIR or polyphase FIR for more information.

Share Improve this answer Follow

answered Oct 26, 2008 at 20:27

Mendelt
37.5k ● 6 ● 75 ● 97

5

Notice (in additions to the other comments) that the simple-easy-intuitive approach "*downsample by a factor of 4 by replacing each group of 4 consecutive samples by the average value*", is not optimal but is nevertheless not wrong, nor practically nor conceptually. Because the averaging amounts precisely to a low pass filter (a rectangular window, which corresponds to a sinc in frequency). What would be conceptually wrong is to just downsample by taking one of each 4 samples: that would definitely introduce aliasing.

By the way: practically any software that does some resampling (audio, image or whatever; example for the audio case: sox) takes this into account, and frequently lets you choose the underlying low-pass filter.

Share Improve this answer Follow

edited May 12, 2010 at 18:23

answered Mar 26, 2010 at 17:34

leonbloy
75.7k ● 21 ● 146 ● 193

1

You need to apply a lowpass filter before you downsample the signal to avoid "aliasing". The cutoff frequency of the lowpass filter should be less than the nyquist frequency, which is half the sample frequency.

Share Improve this answer Follow

edited Oct 26, 2008 at 18:28

answered Oct 26, 2008 at 18:18

Hallgrim
15.5k ● 11 ● 48 ● 54

wouldn't it still introduce noise after downsampling? – artificialidiot Oct 26, 2008 at 18:20

Since the signal is downsampled with a integral factor (44 to 11 khz is 1:4) it will not be too bad. Perfect resampling is an art. – Hallgrim Oct 26, 2008 at 18:28

1

The "best" solution possible is indeed a DFT, discarding the top 3/4 of the frequencies, and performing an inverse DFT, with the domain restricted to the bottom 1/4th. Discarding the top 3/4ths is a low-pass filter in this case.

Padding to a power of 2 number of samples will probably give you a speed benefit. Be aware of how your FFT package stores samples though. If it's a complex FFT (which is much easier to analyze, and generally has nicer properties), the frequencies will either go from -22 to 22, or 0 to 44. In the first case, you want the middle 1/4th. In the latter, the outermost 1/4th.

You can do an adequate job by averaging sample values together. The naïve way of grabbing samples four by four and doing an equal weighted average works, but isn't too great. Instead you'll want to use a "kernel" function that averages them together in a non-intuitive way.

Mathwise, discarding everything outside the low-frequency band is multiplication by a box function in frequency space. The (inverse) Fourier transform turns pointwise multiplication into a convolution of the (inverse) Fourier transforms of the functions, and vice-versa. So, if we want to work in the time domain, we need to perform a convolution with the (inverse) Fourier transform of box function. This turns out to be proportional to the "sinc" function (sin at)/at, where a is the width of the box in the frequency space. So at every 4th location (since you're downsampling by a factor of 4) you can add up the points near it, multiplied by sin (a dt) / a dt, where dt is the distance in time to that location. How nearby? Well, that depends on how good you want it to sound. It's common to ignore everything outside the first zero, for instance, or just take the number of points to be the ratio by which you're downsampling.

Finally there's the piss-poor (but fast) way of just discarding the majority of the samples, keeping just the zeroth, the fourth, and so on.

Honestly, if it fits in memory, I'd recommend just going the DFT route. If it doesn't use one of the software filter packages that others have recommended to construct the filter for you.

Share Improve this answer Follow

answered Oct 26, 2008 at 21:03

wnoise
9,902 ● 39 ● 47

Zeroing DFT coefficients is just like multiplying by a window in the frequency domain. It's not recommended when working with finite length signals. – Royi May 9, 2011 at 14:26

1 @Drazick: It's not "just like", it is. Recommended by who for what purpose, with what justification? If you don't do discard everything with a higher frequency, it shows up aliased at a lower frequency. It's possible that some non-rectangular window that is also zero at and above this frequency gives "smoother" sound, but it then represents the original signal less well. – wnoise May 9, 2011 at 16:37

1

The process you're after called "Decimation". There are 2 steps:

1. Applying Low Pass Filter on the data (In your case LPF with Cut Off at Pi / 4).
2. Downsampling (In you case taking 1 out of 4 samples).

There are many methods to design and apply the Low Pass Filter.

You may start here:

http://en.wikipedia.org/wiki/Filter_design

Share Improve this answer Follow

answered May 9, 2011 at 14:32

Royi
4,943 ● 7 ● 51 ● 72

1

You could make use of libsamplerate to do the heavy lifting. Libsamplerate is a C API, and takes care of calculating the filter coefficients. You to select from different quality filters so that you can trade off quality for speed.

If you would prefer not to write any code, you could just use Audacity to do the sample rate conversion. It offers a powerful GUI, and makes use of libsamplerate for it's sample rate conversion.

Share Improve this answer Follow

answered Jun 5, 2011 at 4:41

Russell
303 ● 1 ● 6

0

I would try applying DFT, chopping 3/4 of the result and applying inverse DFT. I can't tell if it will sound good without actually trying tough.

Share Improve this answer Follow

answered Oct 26, 2008 at 18:27

artificialidiot
5,369 ● 32 ● 27

DFT works in buffers. This introduces all kinds of extra problems. You need to do windowing to eliminate problems at the buffer edges and mix the buffers back togehter with overlap using overlap-save or overlap-add methods. FIR or IIR filters are better. – Mendelt Oct 26, 2008 at 20:30

1 If you can process everything at once, zero padding eliminates all edge issues. – wnoise Oct 26, 2008 at 20:37

If you process everything at once DFT's can take very long. A DFT takes O(n^2). You could use an FFT but that only works if the buffer-size is a power of two. Usually you achieve that by cutting the signal in smaller buffers. – Mendelt Oct 26, 2008 at 21:13

1 FFT is a class of algorithms that is used to calculate DFT. And without zero padding, you get 1/4 of the data which is the point. – artificialidiot Oct 27, 2008 at 11:00

1 Mendelt: FFT is a particular way of calculating the DFT, that is O(n log n) instead O(n^2). It's most easily implemented for powers of two, but it's not restricted to that. Zero padding to a power of two is easy in any case, and no information is lost when doing so. – wnoise Nov 2, 2008 at 18:23

0

I recently came across BruteFIR which may already do some of what you're interested in?

Share Improve this answer

answered Apr 15, 2009 at 11:07

Follow

Jon Cage
37.4k ● 41 ● 145 ● 222

0

You have to apply low-pass filter (removing frequencies above 5500 Hz) and then apply decimation (leave every Nth sample, every 4th in your case).

For decimation, FIR, not IIR filters are usually employed, because they don't depend on previous outputs and therefore you don't have to calculate anything for discarded samples. IIRs, generally, depends on both inputs and outputs, so, unless a specific type of IIR is used, you'd have to calculate every output sample before discarding 3/4 of them.

Just googled an intro-level article on the subject: https://www.dspguru.com/dsp/faqs/multirate/decimation

Share Improve this answer

Follow

edited Mar 16, 2018 at 14:53

Danijel
8,578 ● 19 ● 81 ● 146

answered Jun 24, 2013 at 17:41

noop
1