Acta Cybernetica **19** (2009) 159–175.

Filter Bank Design for Melody Recognition

Zoltán Gera*

Abstract

Recognizing different features of a waveform to later recompose the music that was originally present in the signal is a difficult task. There are numerous fields of application where these techniques are known to be useful including music authoring, digitizer design, automatic music transcription. There are many different methods that can be used for this purpose giving somehow inadequate quality regarding noise, polyphony or time- / frequency localization compared to the human auditory system. In this article, I will show a new filter design method specifically designed to be aware of human perception features. I will also show the way how a complete filter bank can be assembled and used for melody recognition in real time. Finally, I will point out the benefits of this filter design compared to other methods.

Keywords: DSP, melody recognition

1 Introduction

1.1 Main Objectives

Melody recognition is a process where different features are extracted from a waveform to later form music data. These features, such as *pitch*, *note length* and *loudness*, are high-level, subjective and abstract psychological perceptions [9] that are hard to deal with. Recognizing rhythm and melody simultaneously should involve an analyzation process both in time- and frequency domains [14]. This resolution should have special requirements compared to other conventional signal processing techniques. Examining these requirements makes us possible to reach superior quality over standard methods [1, 15, 20].

Time resolution must have an adequate *separation* property. Separation avoids blurring sounds together that were audible as two distinct notes. It should also give *continuity* in a way that it does not leave short but significant sounds out of processing. These requirements, together with their exact parameters, can be derived from psychoacoustic measurements [9].

The frequency resolution should fit the exponential *scale of music* [13]. It should give correct state information about every note along the scale regarding the note

*ELTE IK, Budapest, E-mail: gerazo@elte.hu

is sounded or not. Neighboring notes should be clearly distinguishable from each other. Every sound that has significant harmonic content should be classified along the *melodic scale* as one note. These requirements are related to music theory [13] and psychoacoustics [1, 9, 15].

Our computational load expectancies aim not less than real-time performance. Running the recognition process in real time is a key to give us numerous new ways of utilization creating revolutionary techniques in digital music authoring.

1.2 Idea of Filter Bank

Every note of the melodic scale is now considered as a frequency interval or *channel*. We should create individual filters which fulfill our time domain requirements at one particular channel. These filters do their work with their own channel data only, they do not interfere with others, so the whole bank will also fulfill the frequency domain requirements as well. It is easy to find out that these filters should be *bandpass filters* with a well-defined *passband*. Having the scale of western music [13] where an *octave* difference is a multiplication by 2, an octave has 12 different notes and a convention exists where A-4 note means frequency $440Hz$, we get the following center frequencies for notes:

$$440 \cdot 2^{n/12} \tag{1}$$

where n is the note number relative to A-4. We want to have whole octaves in our range of investigation, so the interval from C-1 to B-9 is $n \in [-45..62]$ which covers the full audible frequency spectrum. That is only 108 filters with the following passbands:

$$\left[440 \cdot 2^{(2n-1)/24}, 440 \cdot 2^{(2n+1)/24}\right] \tag{2}$$

Every filter will be used to measure note state on a channel at a particular point of the input signal. Note state comes from a measurement that gives us some kind of value connected to *perceptual loudness* [9]. The time interval of different measurements should be short enough to fulfill separability criteria. This interval also depends on frequency. The higher the pitch is, the shorter the interval should be. (simple outcome of period length) However, a filter need not to be used at every point of the input data because far less work also gives adequate time domain resolution [8]. It would be also a waste of processing power. This means that only FIR (*Finite Impulse Response*) filters are good for this purpose because IIR (*Infinite Impulse Response*) filters can only be computed along an interval, rather than at a single point, because of their recursive nature. Filter length and every other parameter should be individually computed for every note, and separate filters should be designed for all notes.

Filters should be created to give constant quality on every channel, so delivering the same attenuation features with every note. This idea has the same concept as the classic method called *constant-Q transform* (CQT) [18]. However, CQT and all derived methods [3] lack general using of psychoacoustic principles apart from the fact that they also operate with non-linear scale. This is the explanation why CQT

can be computationally more intensive than FFT, even if CQT produces less data and lacks reversibility features of FFT [18]. Our constructed filters will form a *filter bank*. Under certain circumstances, the bank can produce analysis with adequate quality. I will show the details in the next section.

2 Filter Design

2.1 Creating a Filter

We need a specifically designed bandpass filter which attenuates (suppresses) everything outside passband, but gives perceptual loudness inside. According to the experiments of Fletcher and Munson [5], *equal-loudness contours* (or *Fletcher-Munson curves*) show the characteristics of how the human auditory system responds to different sound pressure levels at different frequencies. To have our filter produce perceptual loudness values rather than absolute levels, the filter's passband should fit the reciprocal of the mentioned curves called *weightings*. Different weighting functions do exist. We will use *A-weighting* in our work because it is designed to work well on harmonic waveforms. Other weightings perform better on noise-like waves. They could be used for special, rhythm intensive recognition tasks.

Using a weighting function to get perceptual loudness instead of simple volume level is a key to be able to later compare different peaks and choose dominant harmonics. It is quite common that the detection or transcription of various instruments give false results because the underlying algorithm uses absolute measurements and direct physical values when dealing with abstract, cognitive parameters. There are more good approximations of the A-weighting. We will use one of them. The ideal filter for one channel and the A-weighting is shown in Figure 1.

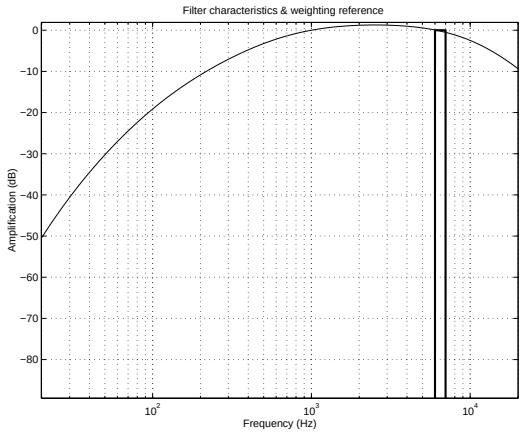

Figure 1: Ideal filter and A-weighting in frequency domain

Creating the impulse response of this filter is easy for any arbitrary note n.

To realize this representation on computer, we choose a number l_d as the *design length* of the filter. Because the filter designing is an off-line procedure (no real-time requirements), we choose l_d to be large. The filter representation is

$$f(x) = \sum_{i=\left\lfloor l_d 440 \cdot 2^{(2n-1)/24}/s \right\rceil}^{\left\lfloor l_d 440 \cdot 2^{(2n+1)/24}/s \right\rceil} \cos\left(\frac{x \cdot 2\pi i}{l_d}\right) \cdot 10^{\frac{w_A(i/l_d \cdot s)}{20}} \tag{3}$$

where f is the vector with length l_d, $x \in [0..l_d - 1]$, the $[\dots]$ operator rounds to the nearest integer, w_A is the A-weighting function and s is the *sample rate* in Hz which defaults to 44100. The multiplication after the cosine applies the weighting function to the result.

One interesting feature of the resulting vector that its significant coefficients are all near the two ends. It can be proven that whenever we add cosines with near frequencies, specially frequencies from a narrow interval, cosine functions are always in phase at the beginning and end of the interval giving the biggest coefficients there. We change the phase of the resulting vector to have the largest coefficients at the center of the vector.

$$f' = \left[f\left(\left\lceil \frac{l_d}{2} \right\rceil\right), \dots, f\left(l_d - 1\right), f\left(0\right), \dots, f\left(\left\lceil \frac{l_d}{2} \right\rceil - 1\right)\right] \tag{4}$$

This allows us to later apply window functions with smaller information loss. The impulse response of this quasi-ideal filter for note n is shown in Figure 2. (The filter is quasi-ideal, because it is discretized now, but is still in long-length representation.)

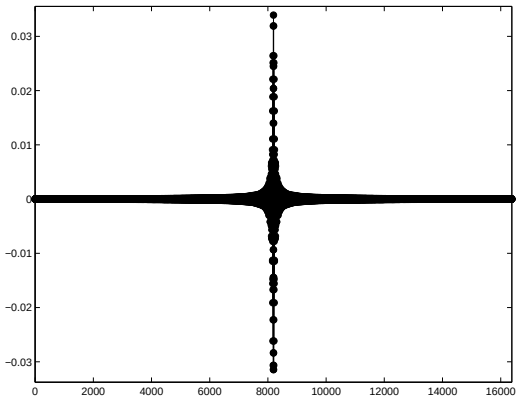

Figure 2: Quasi-ideal filter in time domain

The quasi-ideal filter has still perfect characteristics, but it is far too long to be used in real time. We need to chop insignificant parts off the vector to have a short and real-time applicable filter. Fortunately, we can do this because significant coefficients are in the center of the vector.

Chopping is best done through applying a *window function*. If we use a compact function for this purpose, we can keep the good features previously mentioned. Applying a *window function* on a filter can have further positive effects on the final result. Our goal is to reduce the power of crosstalk coming from outside the passband, so using *Kaiser window* which maximizes the ratio of the mainlobe energy to the sidelobe energy seems to be a good choice. Later we will also determine the parameters of this window. If $g(x)$ is the final filter with length l_f (a parameter to be discussed later), $f'(x)$ is our current filter with length l_d (design length), the window function is $w(x)$ also l_f long, then chopping and applying the window function on filter $f'(x)$ is simply the following

$$f'' = \left[f' \left(\left\lceil \frac{l_d}{2} \right\rceil - \left\lceil \frac{l_f}{2} \right\rceil + 2 \right), \ldots, f' \left(\left\lceil \frac{l_d}{2} \right\rceil + \left\lfloor \frac{l_f}{2} \right\rfloor + 1 \right) \right] \tag{5}$$

$$g(x) = f''(x) \cdot w(x) \tag{6}$$

Filter design through manipulating different window functions is told to be an art, because many times, there are no direct methods of getting the best values. Several experiments are required to find a close-to-the-best solution. This is definitely our case. The previously mentioned windowing step uses a window that is a combination of a standard rectangular window (for chopping) and a Kaiser window (for tuning up characteristics). It is possible to further improve our filter with more window functions combined into our windowing step. No matter how fancy our window composition is, this windowing step is done in one step only at design time. Figure 3 shows the impulse response of the filter after the windowing step.

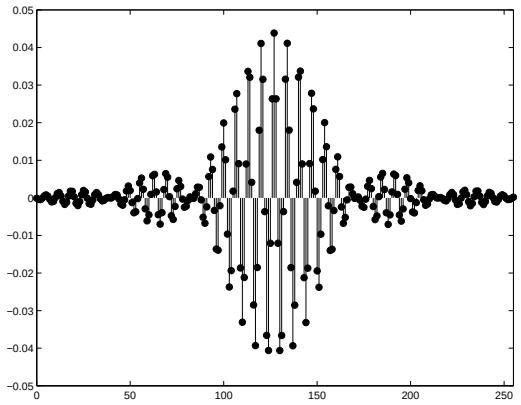

Figure 3: Final impulse response of the filter

Finally, we should normalize the passband response of the filter to have exactly the same attenuation as the A-weighting has at the center frequency of the passband. The easiest way to do this is to use the filter on a sine wave of center

frequency, and measure the SPL (*Sound Pressure Level*) of the resulting wave to have the correct coefficient to be multiplied on the filter. This way, we have gained the desired response properties implicitly compensating the power distortion effect of the windowing step and the filter length difference between different channels.

$$t(x) = \sum_{i=0}^{l_f-1} \cos\left((x-i)\cdot 2\pi \cdot \frac{440\cdot 2^{(2n-1)/24} + 440\cdot 2^{(2n+1)/24}}{s}\right) \cdot g(l_f - i) \quad (7)$$

where g is the unnormalized filter, l_f is the final filter length and the length of g, t is the filtered wave. The length of t should not be too small to have adequate precision. Let this length be l_p (we are still in design time), so variable will go $x \in [0..l_p - 1]$. The SPL of t is the quadratic mean

$$SPL = \sqrt{\frac{\sum_{i=0}^{l_p-1} t(i)^2}{l_p}} \quad (8)$$

so the final filter h will be g multiplied by a constant:

$$h(x) = g(x) \cdot \frac{\sqrt{\frac{1}{2}} \cdot 10^{\left(w_A\left(\frac{440\cdot 2^{(2n-1)/24} + 440\cdot 2^{(2n+1)/24}}{2}\right)/20\right)}}{SPL} \quad (9)$$

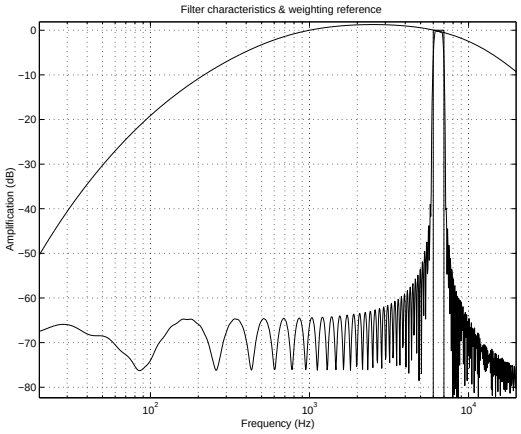

Figure 4: Frequency response of the final filter

Figure 4 shows the frequency response of the final filter. The ripples are the results of giving the filter a short discretized form to fit in a short vector. This is the cost of filtering real-time compared to the ideal filter world.

2.2 Evaluating the Design

Our choice was the Kaiser window as a window function. We can use different β parameters for the window. Zero equals the rectangular window. There are some experiments in Table 1 where different window parameters were used with our design method.

Increasing β lowers sidelobes which reduces crosstalk from stopband, but widens the mainlobe giving smoother transition between stopband and passband. In classical filter design, wider mainlobe is to be avoided. For us, wider mainlobe is not as harmful as crosstalk, because its effects can be reduced by post-processing between neighbors on the final data. This also happens in the human auditory system where auditory nerves apply inhibition on neighboring nerves and cells [15, 16]. This choice is also a key to mimic perceptual functions better.

Evaluating the filter is important. We should know how good it is, what quality does it have. Evaluating a filter is done in frequency domain, in our case, with a logarithmic scale in both directions (frequency and amplitude level). Our design assures good approximation of the weighting function in the passband even in the worst case, so evaluating is mostly necessary in the stopband.

An error function will be created to rate the filter. The stronger the attenuation is in the stopband, the smaller error value the filter will get. There should be an attenuation threshold called the *ideal attenuation* which is adequate to our needs. Beyond this threshold, no error value is given for the filter to the specific frequency point. The errors of different frequency points are cumulated to form the final error value for the filter.

$$e = \sum_{i=1 \& a(i) > a_i}^{p} (a(i) - a_i) \tag{10}$$

where a is the frequency response of the filtering with p precision (length) having the attenuation values for different frequencies. These can be calculated the same way we did at the normalization procedure. The spectrum is calculated only for the audible interval, namely $20Hz - 20kHz$. Coefficient a_i is the ideal attenuation. Setting it to a reasonable value like constant 60 dB gives good results.

As was mentioned before, our design ensures good features in the passband. The frequency response of the whole stopband is quantifically interesting and we use it in our error function. But the accurate frequency response of the stopband has no importance.

This is not the case with the passband. The passband will let through peaks of the original signal which should be later compared to each other perceptually in both time- and frequency domains. The passband for this reason should be strictly fit to the weighting function. This is not a general criterion for all the passbands of different channels, but should specifically apply to all the individual passbands on their own. This causes that even the smallest ripples are not allowed inside the passband. Otherwise, small vibratos or other artistic techniques could cause level change in the output of a filter which was not present in the original signal at all.

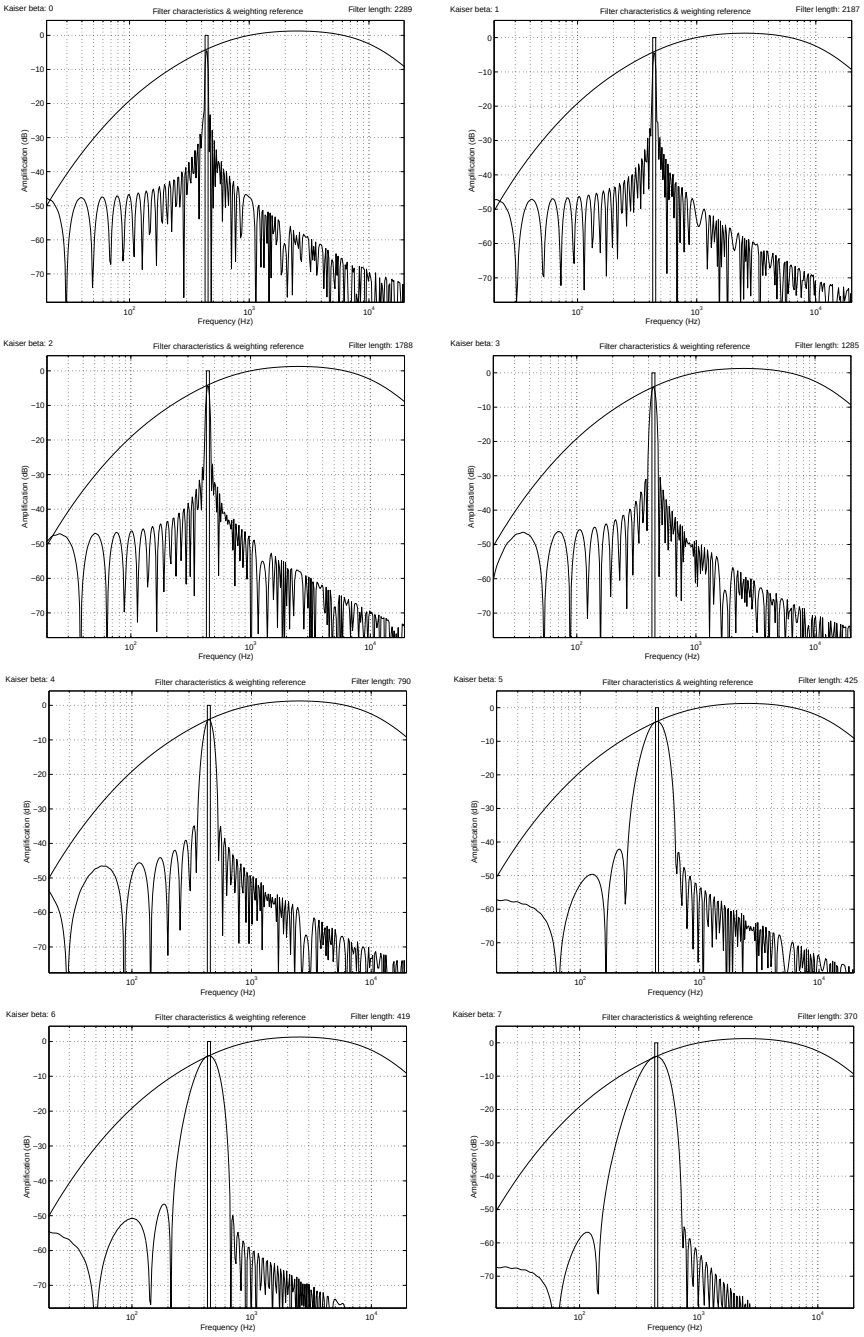

Table 1: Filter design with different Kaiser β parameters

This is a key feature to have a stable and steady output when the note is inside the channel of concern.

The conclusion is that comparing our filter design to other filter design methods has little use. The error function itself also does not serve this purpose. For example, the *Parks-McClellan optimal equiripple* filter design method reduces the number of filter coefficients by introducing ripples in the passband. While this technique turns out to be quantifically better than ours under normal circumstances (evaluating error along a linear scale in both passband and stopband), our method is superior with the mentioned special error function (evaluating along logarithmic scale and only in stopband). Our method introduces ripples only in the stopband reducing the number of coefficients without disturbing the important features gained in the passband.

The length parameter of our filter design is still undetermined, so we get basically random error values after evaluating. Our error function will be used in the following to create an algorithm to determine the optimal filter for every note. This way, we will gain good performing filters on all channels specifically tailored to our needs.

3 Filter Banks

3.1 Filter Bank Assembly

A filter bank consists of many filters. It may contain more than one filter for a note. Our filter design method has many different parameters. We have determined what parameters have optimal values independent of frequency, so these parameters can be the same when designing filters that belong to different notes. However, the most important parameter, the final filter length is still a question. Figure 5 shows different quality measurements of a design of every filter along the scale using constant length parameters.

Obviously, the final filter length parameter cannot be a constant. It will always depend on the note we are designing. The quality measurement of a filter introduced in the previous section is a good point to start from. We should design filters for every note with constant quality (constant error value). We need an algorithm to locate an optimal filter length parameter for every note. Optimality is reached here when the resulting filter's error value is below a preset threshold, and the filter is as short as possible. The pseudo-code of this process should be something like this:

1. Set parameters to their defaults and set length parameter to a really small value
2. Design the filter with current parameters
3. Evaluate the filter (using the error function)
4. If the error is below the quality threshold, but it is close to it, stop procedure

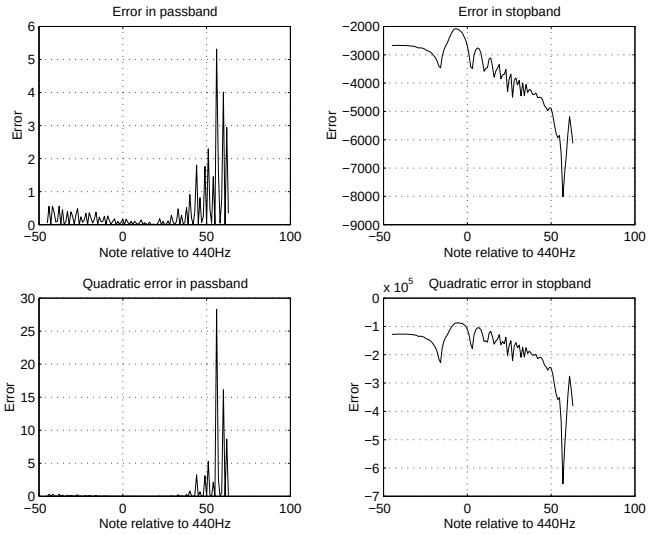

Figure 5: Filter evaluation along the full scale with constant length parameters

5. If the error is smaller then the quality threshold, decrease length parameter by a small amount, go to step 2

6. If the error is bigger then the quality threshold, increase length parameter by a bigger amount, go to step 2

As we have seen, the error values are not monotonic in the function of length. However, the trend of the function is monotonic, only small ripples cause the loss of strict-sense monotonicity. The above algorithm can be implemented as a modified logarithmic search, so determining the exact step amount at the increase/decrease phases is easy. (This exact step amount will be reduced as we get closer according to the original logarithmic search.)

Because this filter design method is not optimized for real time at all (it will run in design time), a search where the design and evaluation phases are repeated many times can be quite time consuming. We do not need to have the best parameter, we only need to get a close approximation. That's the reason of the uncommon stop condition in step 4. The logarithmic search will find a near optimal length value. Choosing smaller steps can improve quality and degrade performance. The bigger step toward longer filters favors quality over length on the long run.

The whole procedure finding optimal filters along the scale with this search algorithm using various error thresholds, extreme small step values and really close stop condition (to reach the ideal optimum and to be able to see the best result with worst performance) runs only some hours long on a common computer (test was done on Celeron 900MHz) in Matlab. This covers running the search algorithm for every channel along the scale multiple times to gain filters for all channels with

different quality parameters (5 filters per channel in this experiment). Because the Matlab testbed was using only basic calculations (own DFT to be able to modify also some parameters inside), recoding the whole procedure in a real programming language should give a drastic speedup to the process. (The same can be run under less than a minute.) Figure 6 shows the lengths of the resulting filters along the scale.

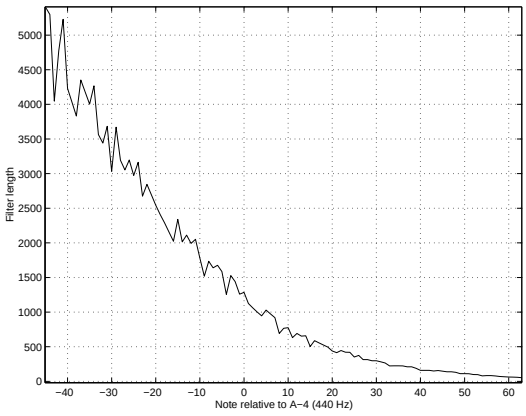

Figure 6: Lengths of filter results

3.2 Filter Bank Usage

Let vector f be the final filter of length l_f for a specific note, and let x be the input waveform. The filtered wave y is then

$$y(i) = \sum_{j=0}^{l_f-1} x(i - l_f + 1 + j) f(j) \tag{11}$$

Our task is to determine the loudness level of the original signal in different points of time via calculating the SPL of y at some points using the smallest number of calculations possible. The above equation allows finding values of y in one arbitrary point. We know that y only contains sines around the center frequency of the note that belongs to the filter. The basic unit of our real-time calculations, the *perceptual loudness measurement* at point i for the filter of note n with s as sample rate is as follows.

$$z(i) = \sqrt{y\left(i - \left\lceil \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rceil\right)^2 + y\left(i + \left\lfloor \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rfloor\right)^2} \tag{12}$$

This is a straightforward consequence of the equation $\sin^2 x + \cos^2 x = 1$.

This measurement is only accurate when we measure a signal containing only the center frequency. Other frequencies can also be present in y. These are frequencies near the center frequency, because we are using a well-defined passband filter. Taking the center as unit, these are $2^{-1/24} \dots 2^{1/24}$ away from the frequency of center which means $-2.85\% \dots 2.93\%$ fluctuation in frequency. This maximum 3% fluctuation level gives the measurement inaccuracy $\sin^2 x + \cos^2 (x + 2\pi \cdot 0.03)$ under the period of the center frequency shown in Figure 7.

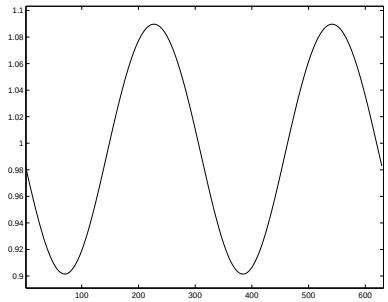

Figure 7: Perceptual loudness measurement inaccuracy through 1 period of center frequency

The fluctuation of inaccuracy is two times faster then the change of the waveform level itself, so it is enough to do the loudness measurement once per period changing the exact phase a bit randomly, and average the last measurements to suppress the fluctuation and get the real perceptual loudness level. According to psychoacoustic considerations, we even do not have to do the measurement once per period. It is adequate to do it rarelier, but still the random phase change and averaging logic should be used.

Summing up the consequences, we only have to do the loudness measurement rarely, at most once per period (or rarelier if psychoacoustic features allow this). This only takes using the filter twice to calculate two distinct points of y. The function y is never fully computed. The random phase change in choosing the discrete points of y to be computed is not a simple smoothing technique. It is necessary to compensate for slightly out of tune instruments, but first of all, to compensate for inharmonic content of instruments. Dealing with inharmonic content is the most important factor of this kind of smoothing, because inharmonicity is there even in the most harmonic real-world instruments. Inharmonicity is also an organic part of rhythm instruments.

It is obvious, that the measurement rate is proportional to the center frequency, so the note to which the filter belongs. Figure 8 shows the ratio between the filter length and the period length of the center frequency.

This result is really interesting, because not only does it show that the longer filter length at low frequencies are compensated by the fact that these filters should be applied less, but it also turns out that using the method is cheaper on lower

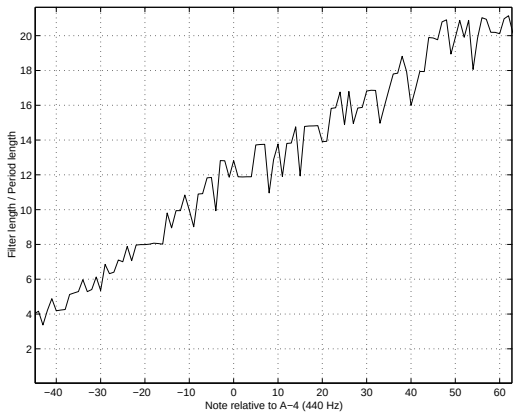

Figure 8: Filter length per period length ratio on scale

notes.

It is possible to construct more filter banks with different quality threshold values. This would give the possibility to the real-time method to use extra computational power on better machines, and still give adequate results on low-end ones. It is also possible to dynamically and adaptively change the used filter bank to have good time- and frequency resolution as well. This trick can be used to virtually get rid of the Heisenberg uncertainty principle similar to what happens in the human brain as supposed by biologists [17].

3.3 Comparing to Other Methods

Finally, we do some comparisons. First, we compare our method to a trivial solution of our problem that uses FFT. Special comparing is needed in case of real-time applications. A method running in real time is better described by the number of operations per second than the asymptotic number of operations in the function of input data (especially because the input rate is constant). However, our calculations will still be estimations under certain circumstances. We will investigate the basic case of the two algorithms, but we will focus on comparing the two with parameters having similar quality features. This quality will be really high to be able to see the significant differences, if there are any.

We start with our method. Using the filter on one point according to

$$y(i) = \sum_{j=0}^{l_f-1} x(i - l_f + 1 + j) f(j) \tag{13}$$

takes

$$2 \cdot l_f \tag{14}$$

operations. A perceptual loudness measurement at a point according to

$$z(i) = \sqrt{y\left(i - \left\lceil \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rceil\right)^2 + y\left(i + \left\lfloor \frac{s}{440 \cdot 2^{n/12}} \cdot \frac{1}{8} \right\rfloor\right)^2} \tag{15}$$

takes

$$4 \cdot l_f + 8 \tag{16}$$

operations.

Figure 6 gives us the exact $l_f(n)$ filter length values for every note. We also know the center frequency $f_c(n)$ of note n. For simplicity, we use only one filter bank, but we make a measurement on every period of the center frequency. This gives the total number of operations per second as follows:

$$\sum_{n=-45}^{63} f_c(n) \cdot (4l_f(n) + 8) \approx 250,475,000 \tag{17}$$

This 251 million operations per second need a high-end machine to accomplish. Of course, the 60dB attenuation threshold can be lowered to give still adequate results with relaxed computational needs.

Now, we create a similar method using FFT to reach the same high quality resolution. We use a simple method of FFT overlapping windows and calculate the number of operations for this method. For simplicity, we omit the use of window functions (which is obviously a cheat to help FFT).

The center frequency of the two lowest notes are $32.703Hz$ and $34.648Hz$, therefore the minimal required resolution of FFT is approximately $2Hz$. At sample rate $44100Hz$, this means a window length of 22050. The first applicable window length for FFT is therefore $2^{15} = 32768$. Using conventional radix-2 FFT with bit reversal in the beginning, the number of operations for one FFT is

$$32,768 + 15 \cdot (5 \cdot 32,768) = 2,490,368 \tag{18}$$

To reach the desired time resolution as well, separate FFTs should be calculated according to the center frequency of the highest note. This is $16,744Hz$, therefore the total number of operations per second is

$$2,490,368 \cdot 16,744 = 41,698,721,792 \tag{19}$$

and the ratio between the two methods is

$$\frac{41,698,721,792}{250,475,000} \approx 166 \tag{20}$$

Our algorithm is significantly faster than trivial FFT algorithm for the same task. In the investigation above, we have set all parameters to get the same quality from both methods. FFT turned out to be much slower. This also means, that with the same performance (same number of operations), FFT gives much lower quality

(which is inadequate for our special task) or on the other hand, our new algorithm delivers far higher quality than FFT with the same number of operations.

Our special requirement is a frequency resolution using logarithmic scale. Using wavelet-like methods introduced in [14] would give high performance with extreme poor quality, because the octave frequency resolution is far not enough for a 12-degree scale (Wavelet transforms are reversible, so dividing an octave into 12 parts would be a waste according to the wavelet philosophy).

Methods that use linear scale can deliver adequate quality, but perform worse than ours. This means lower quality with same performance (number of operations), or much lower performance with the same quality, simply because linear scale does not fit our task. This is the case using FFT (as shown above) and FFB according to [3]. FFB (*Fast Filter Bank*) uses basically the same frequency resolution as FFT does, but uses more sophisticated filters for tuning up individual channel characteristics. The cost is also more operations per second which further lowers performance.

The original CQT introduced in [18] uses pure logarithmic frequency resolution fit to the music scale. The strange property of CQT is that it needs far more computations than FFT according to the original [18] results. There are more papers about faster methods of calculating CQT, the most recent is [3], but none of them could be faster than FFT. This is the simple result of CQT's original concept, that it wants to deliver continuous signal output as FFT does. While this can be useful for some special off-line analyzation purposes (examining classical big orchestral music in printed paper), it was shown above that general melody recognition does not need this feature as the human brain also lacks this huge precision of resolution. My method can be faster than CQT only because I use psychoacoustic features which are not used by previous works.

The other remarkable result of [3] is the CQFFB (*Constant Quality Fast Filter Bank*), which also uses the filter tuning trick on CQT to deliver better separation properties. While this further lowers performance but raises quality compared to CQT, the main problem with this is that the individual filters are still not constructed according to the features of music perception. They rather form a consistent bank of filters with similar passband limits, but passband frequency responses are not designed intentionally. CQFFB also lacks deep psychoacoustical design so inherits the weaknesses of CQT regarding low performance (high number of operations).

It should be also mentioned that both CQT and CQFFB [3] methods have accelerated versions (BQT and BQFFB). These *bounded-Q* methods gain performance by using only one bank for an octave instead of using banks for all channels as the original versions did. This trick makes the frequency separation worse. Different notes have slightly different filters and passbands (because inside an octave, the frequency resolution is linear), so it is impossible to design filters with as strict criteria as we did. My method designs filters specifically for all individual notes, and only one filterbank is constructed which can be applied much faster delivering the expected frequency responses.

References

[1] Aslan, A. Music Perception as a Topic of Cognitive Psychology. Doguş Universitesi Dergisi. 8(2):117-127, 2007.

[2] Bohn, D. *Audio Specifications.* Rane Corporation, 2000.

[3] Diniz, F. C. C. B., Kothe, I., Netto, S. L., Biscainho, L. W. P. High-Selectivity Filter Banks for Spectral Analysis of Music Signals. *EURASIP Journal on Advances in Signal Processing*, Volume 2007:1-12, 2007.

[4] Fitch, J., Shabana, W. A Wavelet-based Pitch Detector for Musical Signals. Department of Mathematical Sciences, University of Bath, 1999.

[5] Fletcher, H., Munson, W. A., Loudness, its definition, measurement and calculation. *Journal of the Acoustical Society of America*, vol.5, 1933.

[6] Gera, Z. *Dallamfelismerés és kottázás valós időben.* Master Thesis, ELTE, 2004.

[7] Hacihabiboglu, H., Canagarajah, N. Instrument Recognition Based Wavelet Packet Trees in Audio Feature Extraction. *International Symposium on Musical Acoustics*, Digital Music Research Group, Department of Electrical and Electronic Engineering, University of Bristol, 2001.

[8] He, X., Scordilis, M. S. Psychoacoustic Music Analysis Based on the Discrete Wavelet Packet Transform. *Research Letters in Signal Processing*, Volume 2008:1-5, 2008.

[9] Iakovides, S. A., Iliadou, V. T., Bizeli, V. T., Kaprinis, S. G., Fountoulakis, K. N., Kaprinis, G. S. Psychophysiology and psychoacoustics of music: Perception of complex sound in normal subjects and psychiatric patients. *Annals of General Hospital Psychiatry*, 3:6:1-4, 2004.

[10] Jun, Z., Lei, G., Deyun, Z. A High-performance Psychoacoustics Approach to Speech Quality Evaluation. *Information Technology Journal*, 5(3): 485-488, 2006. Asian Network for Scientific Information

[11] Liu, K. J. R., Wu, A., Raghupathy, A., Chen, J. Algorithm-Based Low-Power and High-Performance Multimedia Signal Processing. IEEE, 1997.

[12] Mannell, R.H., The perceptual and auditory implications of parametric scaling in synthetic speech. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1994.

[13] Mendel, A. Pitch in Western Music since 1500. A Re-Examination. *Acta Musicologica*, 1978.

[14] Meyer, Y. *Wavelets, Algorithms & Applications.* Society for Industrial and Applied Mathematics, Philadelphia, 1993.

[15] Obleser, J., Elbert, T., Eulitz, C. Attentional influences on functional mapping of speech sounds in human auditory cortex. *BMC Neuroscience*, 5:24:1-9, 2004.

[16] Okamoto, H., Kakigi, R., Gunji, A., Pantev, C. Asymmetric lateral inhibitory neural activity in the auditory system: a magnetoencephalographic study. *BMC Neuroscience*, 8:33:1-6, 2007.

[17] Poeppel, D. The Analysis of Speech in Different Temporal Integration Windows: Cerebral Laterization as Assymetric Sampling in Time. *Speech Communication* Volume 41, Issue 1, 2003.

[18] Brown, J. C., Puckette, M. S. An efficient algorithm for the calculation of a constant Q transform. *Journal of the Acoustical Society of America*, vol. 92, no. 5, 1992.

[19] Slaney, M., Lyon, R. F. A Perceptual Pitch Detector. *International Conference on Acoustics Speech and Signal Processing*, 1990.

[20] Todt, D. From birdsong to speech: a plea for comparative approaches. *Anais da Academia Brasileira de Ciencias*, 76(2):201-208, 2004.

Received 22nd August 2007