PAPER *Special Section on Latest Advances in Fundamental Theories of Signal Processing*

Fast and Accurate Generalized Harmonic Analysis and Its Parallel Computation by GPU

Hisayori NODA[†a)], ***Nonmember and*** **Akinori NISHIHARA**[†], ***Member***

SUMMARY A fast and accurate method for Generalized Harmonic Analysis is proposed. The proposed method estimates the parameters of a sinusoid and subtracts it from a target signal one by one. The frequency of the sinusoid is estimated around a peak of Fourier spectrum using binary search. The binary search can control the trade-off between the frequency accuracy and the computation time. The amplitude and the phase are estimated to minimize the squared sum of the residue after extraction of estimated sinusoids from the target signal. Sinusoid parameters are recalculated to reduce errors introduced by the peak detection using windowed Discrete-Time Fourier Transform. Audio signals are analyzed by the proposed method, which confirms the accuracy compared to existing methods. The proposed algorithm has high degree of concurrency and is suitable to be implemented on Graphical Processing Unit (GPU). The computational throughput can be made higher than the input audio signal rate.

key words: *generalized harmonic analysis, graphical processing unit*

1. Introduction

Generalized harmonic analysis (GHA) [1] is a concept of signal analysis introduced by N. Wiener in 1930, in which a target signal is expressed as sum of sinusoids. Each sinusoid has three parameters; frequency, amplitude, and phase. Unlike short-time Fourier Transform, the frequency is not restricted to multiples of inverse of the frame size. So the frequency resolution is very high by its nature, and the time resolution is also high because the frame size can be made short without affecting the frequency resolution.

A target signal is divided into frames having size N. The signal in a frame is approximated by the sum of sinusoids as

$$x_0(n) \cong \sum_{k=1}^{K} A_k \sin(\omega_k n + \phi_k), \tag{1}$$

where A_k, ω_k and ϕ_k are the amplitude, the angular frequency and the phase of the k-th sinusoid, respectively, and K is the number of sinusoids to be extracted. We can easily synthesize the signal using the sinusoidal parameters. We estimate these parameters so as to minimize the difference between the target signal and the synthesized signal.

GHA is used for several applications. Hirata [2] and Nakazawa [3] applied GHA to the audio coding. Hirata's method achieved high compression ratio for speech audio. Nakazawa's method achieved high quality and high speed compression using $1/12N$ octave frequency quantization for sounds of musical instruments.

Takamizawa [4] proposed a method to repair sound on a SP record. The method reduces pulse noises using Wavelet transform as preprocessing, extracts important sound elements using GHA and resynthesizes the target signal.

A disadvantage of GHA has been its computational cost, which limits the applications of GHA. In this paper, we propose a fast and accurate algorithm for GHA using a peak detection of Fourier spectrum, binary search around that peak to estimate the frequency parameters. The other parameters are estimated by minimizing residual signal power.

The algorithm can calculate GHA much faster than conventional methods, and can be made even faster using high degree of concurrency in the algorithm. When the algorithm is implemented on GPU, the computational throughput is higher than the input audio signal rate.

2. GHA

2.1 ABS Method

George and Smith proposed an algorithm for GHA named Analysis By Synthesis (ABS) [5], in which sinusoids are extracted one after another as

$$x_k(n) = x_{k-1}(n) - S_k(n), \quad k = 1, 2, \cdots \tag{2}$$

where $S_k(n)$ is the k-th sinusoid given by

$$S_k(n) = A_k \sin(\omega_k n + \phi_k). \tag{3}$$

Sinusoid parameters are found by linear search to minimize the power of the residue x_k defined as

$$E_k = \sum_{n=0}^{N-1} \{x_k(n)\}^2, \tag{4}$$

where N is the frame size. Once parameters of the k-th sinusoid are estimated, that sinusoid is extracted from $x_{k-1}(n)$ as in (2). This process is repeated K times to approximate the original signal as

$$x_0(n) \cong x_K(n) = \sum_{k=1}^{K} A_k \sin(\omega_k n + \phi_k). \tag{5}$$

This algorithm takes much time because multiple parameters are searched simultaneously by linear search.

Manuscript received July 4, 2008.
Manuscript revised October 17, 2008.

[†]The authors are with the Department of Communications and Integrated Systems, Tokyo Institute of Technology, Tokyo, 152-8550 Japan.
a) E-mail: hisayori@nh.cradle.titech.ac.jp

DOI: 10.1587/transfun.E92.A.745

2.2 Ushiyama's Algorithm

In 1994, another algorithm for GHA [6]–[8] was proposed, in which the template function is defined by

$$S_k(n) = A_k \sin\left(\frac{2\pi n}{T_k}\right) + B_k \cos\left(\frac{2\pi n}{T_k}\right). \tag{6}$$

The period of the k-th sinusoid T_k is linearly searched with an appropriate step size. The m-th searched point of T_k is referred to as $T_k^{(m)}$, and the corresponding amplitudes of sine and cosine are calculated by

$$A_k^{(m)} = \frac{2}{rT_k^{(m)}} \sum_{n=0}^{rT_k^{(m)}-1} x_{k-1}(n) \sin\left(\frac{2\pi n}{T_k^{(m)}}\right), \tag{7}$$

$$B_k^{(m)} = \frac{2}{rT_k^{(m)}} \sum_{n=0}^{rT_k^{(m)}-1} x_{k-1}(n) \cos\left(\frac{2\pi n}{T_k^{(m)}}\right), \tag{8}$$

where r is the largest integer which satisfies $rT \leq N$. Among all the searched points, parameters which minimize the squared sum of the residual error

$$E_k^{(m)} = \sum_{n=0}^{N-1} \{e_k^{(m)}(n)\}^2, \tag{9}$$

$$e_k^{(m)}(n) = x_{k-1}(n) - \left\{A_k^{(m)} \sin\left(\frac{2\pi n}{T_k^{(m)}}\right) + B_k^{(m)} \cos\left(\frac{2\pi n}{T_k^{(m)}}\right)\right\}, \tag{10}$$

are adopted as A_k, B_k and T_k.

Since this algorithm searches only $T_k^{(m)}$ and amplitudes are determined by (7) and (8), it is faster than ABS method.

2.3 Hirata's Algorithm

In 1998, Hirata proposed a fast algorithm for GHA [2], [9], in which FFT spectrum of the target signal is first used to find a rough estimate of the frequency. To estimate the frequency of the k-th sinusoid, the peak frequency of the FFT spectrum of the previous residue $x_k(n)$ is taken as the initial value and named as $f_k^{(0)}$. In the m-th search phase, the frequency is searched in the region between $f_k^{(m-1)} - F_s/(2^m N)$ and $f_k^{(m-1)} + F_s/(2^m N)$, and the frequency which minimizes the squared sum of the residue is chosen as $f_k^{(m)}$.

This method can reduce the number of search while keeping the accuracy of the estimated frequency.

3. Improvement of Parameter Estimation

In this section, we propose a method to estimate sinusoidal parameters to be extracted from a target signal. Our method applies peak detection of the Fourier spectrum to estimate parameters and the frequency accuracy is improved at every iteration.

3.1 Frequency Estimation

In the first step, we estimate the frequency (angular frequency) of a sinusoid to be extracted. In this step, we use discrete time fourier transform (DTFT) of the frame defined by

$$X_k(\omega) = \sum_{n=0}^{N-1} x_k e^{-i\omega n}. \tag{11}$$

We search ω which maximizes $|X_k(\omega)|$ defined by

$$|X_k(\omega)| = \sqrt{X_{k_r}^2(\omega) + X_{k_i}^2(\omega)} \tag{12}$$

$$X_{k_r}(\omega) = \sum_{n=0}^{N-1} x_k(n) \cos(\omega n) \tag{13}$$

$$X_{k_i}(\omega) = \sum_{n=0}^{N-1} x_k(n) \sin(\omega n), \tag{14}$$

where $X_{k_r}(\omega)$ and $X_{k_i}(\omega)$ are the real and imaginary part of (11).

Instead of DTFT, we apply FFT to a target signal to get sampled frequencies. Among FFT spectra a peak is detected and call it $\omega(0)$. The truly optimal ω is considered to be around $\omega(0)$. So the range from $\omega(0) - 2\pi/N$ to $\omega(0) + 2\pi/N$ is searched to find the optimal ω, where N is the frame size.

We assume $|X_k(\omega)|$ is convex upward in this range so that

$$\frac{\partial |X_k(\omega)|}{\partial \omega} = 0 \tag{15}$$

holds at the maximum ω.

The derivative is calculated as

$$\begin{aligned} \frac{\partial}{\partial \omega}|X_k(\omega)| &= \frac{\partial}{\partial \omega}\sqrt{X_{k_r}^2(\omega) + X_{k_i}^2(\omega)} \\ &= \frac{X_{k_r}(\omega)\frac{\partial}{\partial \omega}X_{k_r}(\omega) + X_{k_i}(\omega)\frac{\partial}{\partial \omega}X_{k_i}(\omega)}{\sqrt{|X_k(\omega)|}} \\ &= 0, \end{aligned} \tag{16}$$

where $\frac{\partial}{\partial \omega}X_{k_r}(\omega)$ and $\frac{\partial}{\partial \omega}X_{k_i}(\omega)$ are expressed as

$$\frac{\partial}{\partial \omega}X_{k_r}(\omega) = -\sum_{n=0}^{N-1} n x(n) \sin(\omega n) \tag{17}$$

$$\frac{\partial}{\partial \omega}X_{k_i}(\omega) = \sum_{n=0}^{N-1} n x(n) \cos(\omega n). \tag{18}$$

$\frac{\partial}{\partial \omega}|X_k(\omega)|$ is a monotonically decreasing function in this range, and ω can be found by binary search. We can narrow the search range by $1/2$ in one iteration. So calculation accuracy of ω is 2^{-M} where M is the number of iteration. In other words, we can estimate ω in time complexity $O(-N \log \varepsilon)$ where ε is tolerance of frequency.

When we search ω by binary search, the numerator of

(16) is unimportant, because only the sign of (16) is considered in the binary search.

Pseudo code to search ω is shown below. `searchOmega()` returns a value of ω searched by binary search, where `g` is $\omega(0)$ detected by FFT, `N` is the frame size and `x` are samples of the frame. `grad()` returns the denominator of (16), where `o` is ω of (16)

```
float grad(float o, float x[])
{
  float a = 0;
  float b = 0;
  float da = 0;
  float db = 0;
  for (int n = 0; n < N; ++n) {
    float s = sin(o * n);
    float c = cos(o * n);
    a += x[n] * s;
    b += x[n] * c;
    da += n * x[n] * c;
    db -= n * x[n] * s;
  }
  return a * da + b * db;
}

float searchOmega(float g, int N, float x[])
{
  float l = g - 2.0f * PI / N;
  float r = g + 2.0f * PI / N;
  for (int i = 0; i < ITERATION; ++i) {
    float m = (l + r) * 0.5f;
    if (grad(m, x) > 0) {
      r = m;
    } else {
      l = m;
    }
  }
  return (l + r) * 0.5f;
}
```

The searched ω is the k-th frequency ω_k.

This algorithm may not work well when there are more than two local maxima of $|X_k(\omega)|$ in the range from $\omega(0) - 2\pi/N$ to $\omega(0) + 2\pi/N$. Otherwise it works well even if some noises, Gaussian noise for example, are mixed in the target signal. In that case, not only sinusoids which construct the target signal but also noise components are extracted.

The time complexity of this method is the same as that of Hirata's algorithm. But the number of calculation is about 1/4 of Hirata's algorithm, because the number of sum calculation in $|X_k(\omega)|$ is half of the A_k and B_k calculations in Hirata's method, and the iteration of the binary search is half of the search routine of Hirata's method.

3.2 Phase and Amplitude Estimation

After estimation of the frequency, we can estimate the phase of the sinusoid simply by

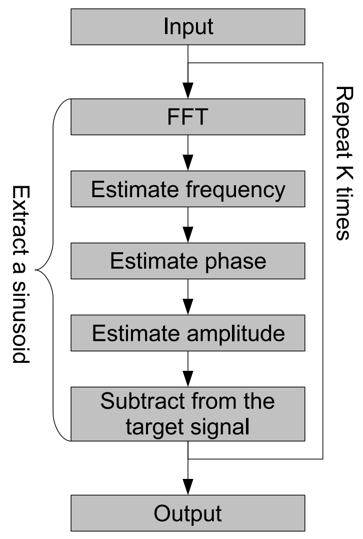


Fig. 1 Algorithm flow of the proposed method.

$$\phi_k = \arctan\left(\frac{X_{k_i}(\omega_k)}{X_{k_r}(\omega_k)}\right). \tag{19}$$

To estimate the amplitude of the sinusoid, we use the square sum of the residue. We estimate the amplitude to minimize E_k, expressed by

$$E_k = \sum_{n=0}^{N-1}\{e_k(n)\}^2 \tag{20}$$

$$e_k(n) = x_{k-1}(n) - A_k \sin(\omega_k n + \phi_k). \tag{21}$$

To minimize it, we calculate $\frac{\partial E_k}{\partial A_k} = 0$. It is derived as

$$\begin{aligned}\frac{\partial E_k}{\partial A_k} &= \frac{\partial}{\partial A_k}\sum_{n=0}^{N-1}\{x_{k-1}(n) - A_k \sin(\omega_k n + \phi_k)\}^2 \\ &= \sum_{n=0}^{N-1}\{2(x_{k-1}(n) - A_k \sin(\omega_k n + \phi_k)) \\ &\quad \sin(\omega_k n + \phi_k)\} = 0\end{aligned} \tag{22}$$

$$A_k = \frac{\sum_{n=0}^{N-1} x_{k-1}(n)\sin(\omega_k n + \phi_k)}{\sum_{n=0}^{N-1}\sin^2(\omega_k n + \phi_k)}. \tag{23}$$

After calculating parameters of the sinusoid, we subtract it from the target signal. We repeat these steps until enough number of sinusoids are extracted.

This entire algorithm is shown in Fig. 1.

3.3 Time Complexity

Time complexities for FFT, frequency estimation, phase estimation, amplitude estimation and subtraction are $O(N\log N)$, $O(-N\log\varepsilon)$, $O(1)$, $O(N)$ and $O(N)$, respectively. We repeat sinusoid extraction for K times. So the time complexity of the proposed method is

$$O(K(N\log N - N\log\varepsilon + 1 + N + N))$$

$$= O\left(NK \log \frac{N}{\varepsilon}\right). \tag{24}$$

The time complexity of Hirata's algorithm is also $O(NK \log \frac{N}{\varepsilon})$. But the number of calculations is four times bigger than our proposed method.

4. Recalculation of Extracted Sinusoids

To speed up the calculation we used windowed DTFT whose samples are actually computed by FFT. This windowed DTFT naturally introduces an effect of the (rectangular) window, which appears as sidelobes of spectra. Those sidelobes violate the accurate estimation of sinusoidal parameters.

To avoid this problem, the frequency of the extracted sinusoid is recalculated after other sinusoids are extracted. Since GHA is basically undisturbed by the window, we can recalculate the sinusoidal parameters using the ones obtained in the previous section as good approximate values.

We propose two recalculation algorithms.

4.1 Single Recalculation

The first algorithm recalculates the parameters of the extracted sinusoids one by one.

Before a new sinusoid is extracted, the parameters of the extracted sinusoids are recalculated. The effect of the sidelobe becomes bigger in proportion to the amplitude of the sinusoid. So the parameters of the first extracted sinusoid, which has the biggest amplitude, are recalculated first to have less effect to the calculation error of the other sinusoids, and then gradually smaller sinusoids are recalculated.

The k-th extracted sinusoid is added back to the residual signal to recover $x_{k-1}(n)$. A sinusoid is extracted from $x_{k-1}(n)$ again by the method in Sect. 3 to replace the sinusoidal parameters. This process is repeated until all the sinusoids are recalculated.

We call this algorithm as single recalculation algorithm, and Fig. 2 shows its flow.

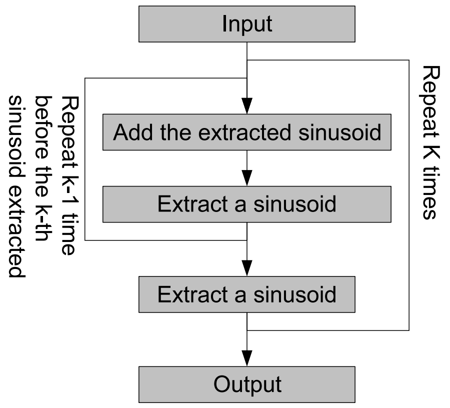

Input
Add the extracted sinusoid
Extract a sinusoid
Extract a sinusoid
Output
Repeat k-1 time before the k-th sinusoid extracted
Repeat K times

Fig. 2 Recalculating one by one.

4.2 Double Recalculation

The second algorithm recalculates the parameters of two sinusoids adjacent in the frequency domain at the same time, to remove possible redundancy where two sinusoids have the identical frequency. This problem arises because the amplitude estimation is not optimal. That is also due to inaccuracy introduced by the windowed DTFT, which is corrected at this stage.

The algorithm is the same as the previous subsection except the following steps. Before recalculation the extracted sinusoids are sorted by their frequencies, which clarifies the existence of two identical frequencies. Possible two identical sinusoids are combined, added back to the residual signal, and the sinusoid of that frequency is recalculated just like the single recalculation algorithm. This algorithm reduces the redundancy and thus improves the overall accuracy.

We call this method as double recalculation algorithm. Figure 3 shows the algorithm flow.

4.3 Time Complexity

Time complexities of FFT, frequency estimation, phase estimation, amplitude estimation, addition and subtraction are $O(N \log N)$, $O(-N \log \varepsilon)$, $O(1)$, $O(N)$, $O(N)$ and $O(N)$, respectively. We repeat sinusoid extraction for $\frac{1}{2}K^2$ times. So the overall time complexity of the proposed method is

$$O(K^2(N \log N - N \log \varepsilon + 1 + N + N + N))$$
$$= O\left(NK^2 \log \frac{N}{\varepsilon}\right). \tag{25}$$

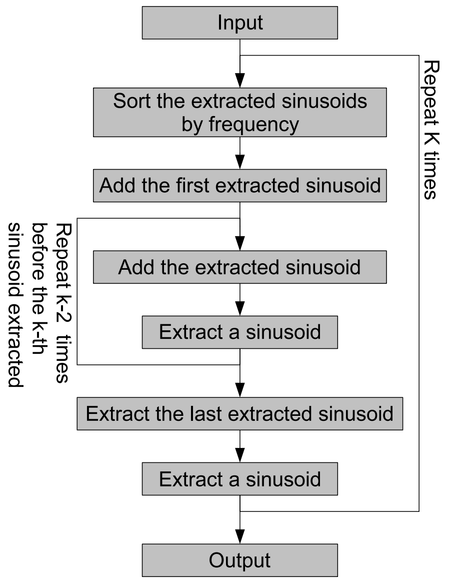

Input
Sort the extracted sinusoids by frequency
Add the first extracted sinusoid
Add the extracted sinusoid
Extract a sinusoid
Extract the last extracted sinusoid
Extract a sinusoid
Output
Repeat k-2 times before the k-th sinusoid extracted
Repeat K times

Fig. 3 Recalculating two by two.

5. Parallel Calculation by GPU

Graphics Processing Units (GPUs) are used for general purpose computations these days. In this case they are called General Purpose GPUs (GPGPUs) [10]. The proposed method in this paper computes a given signal in frames, and each frame is independent from other frames. That is, the algorithm has high degree of parallelism, and it is suitable for parallel computation.

The proposed method is computed using GPU simply by allocating each frame computation to a computational unit of GPU.

5.1 Implementation

CUDA is an environment to support programmers to code algorithms for execution on GPU. In CUDA (compute unified device architecture) [11], the computation programs are divided into small programs which are called "kernel program" and kernel programs are calculated by GPU. The proposed method can be divided into five kernel programs as

INITIALIZATION Initialize the target signal data
FFT Apply Fast Fourier Transform to the target signal
EXTRACTION Extract a sinusoid and subtract it from the target signal (or residue in the following cycles)
ADDITION Add the sinusoid back to the target (residue) signal
SORT Sort the extracted sinusoids by those frequencies,

where FFT is computed by CUFFT library included in CUDA.

In the first step, INITIALIZATION program is run to locate the target signal data to the device memory on the GPU. Then FFT, EXTRACTION, ADDITION and SORT programs are run on GPU according to the order shown in Figs. 1, 2 and 3.

After the whole computations on GPU, the resultant data are copied from the GPU memory to the host memory, which are our results.

The programs except INITIALIZATION program are completed on GPU and do not transfer data between the host and GPU during the computation, and this computation-intensive nature of the proposed algorithm is quite suitable for GPU computation.

6. Experiment

6.1 Experimental Method

We implemented the proposed methods using C++ and CUDA, and measured the extraction accuracy and the computational time. Ten-second, 44.1 kHz sampled, 16-bit, stereo, rock music was used as the target signal. Figure 4 shows Fourier spectrum of the target signal. We extracted 128 sinusoids from each of 512-point frames of the target signal, resynthesized them and measured the Generalized

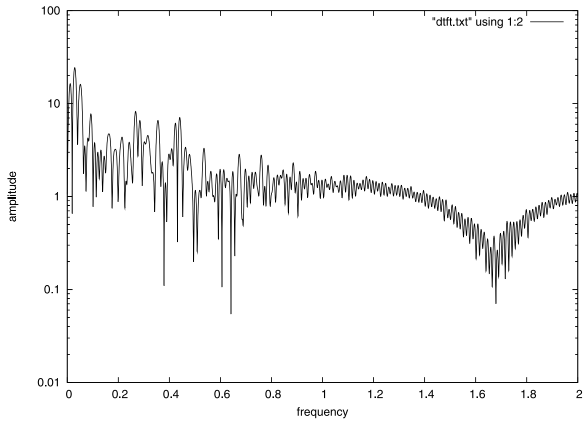


Fig. 4 Fourier spectrum of the target signal.

Table 1 Machine environment.

CPU	Intel Core 2 Quad Q6600
MEM	2.0 GByte
HDD	80 GByte
OS	Windows Vista 64-bit
IDE	Visual Studio 2008

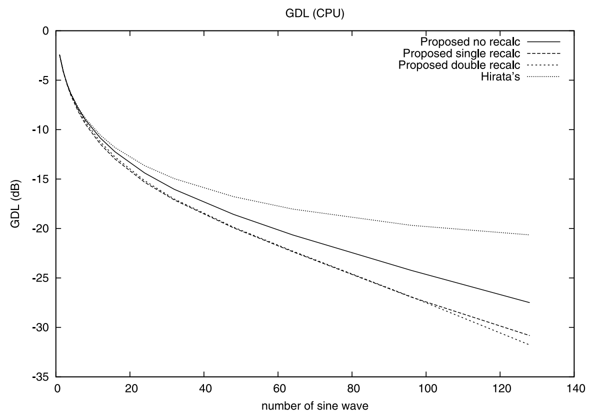


Fig. 5 GDL of resynthesized sinusoids computed by the CPU.

Distortion Level (GDL) [9] between the target signal and the resynthesized signals, where the repeat count of binary search is 20 times. The GDL is defined by

$$GDL = 10\log_{10}\frac{\sum_{n=0}^{N-1}\{x_0(n)-\hat{x}_0(n)\}^2}{\sum_{n=0}^{N-1}\{x_0(n)\}^2}\ [\text{dB}], \qquad (26)$$

where $\hat{x}_0(n)$ is the resynthesized signal.

Microsoft Visual C++ 9.0 (MSVC9) and Intel C++ Compiler 10.1 (ICC10) are used for the C++ program and CUDA 2.0 beta 1 is used for the CUDA program as compilers. Table 1 shows the specifications of a machine used for our experiment.

6.2 Results

Figures 5–13 and Table 3 show the results of the experiments.

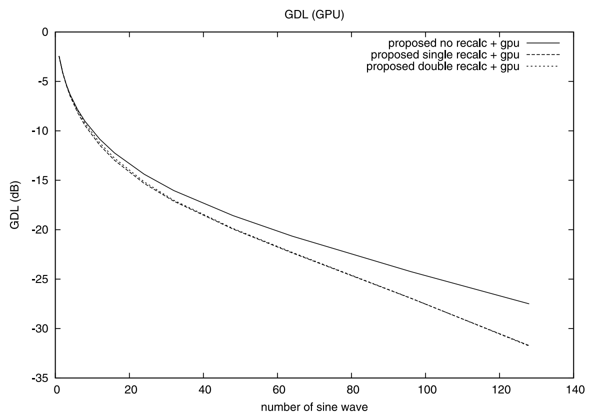


Fig. 6 GDL of resynthesized sinusoids computed by the GPU.

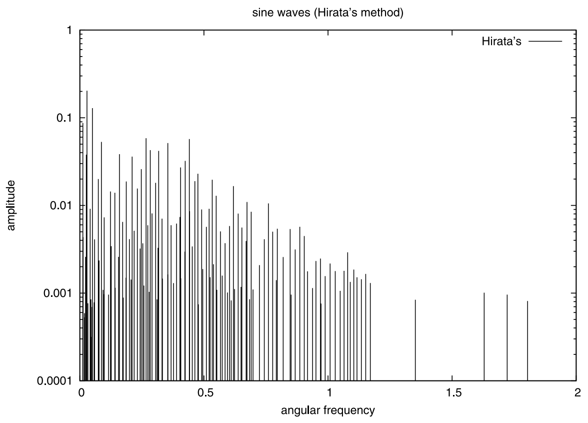


Fig. 7 Extracted sinusoids computed by the Hirata's method.

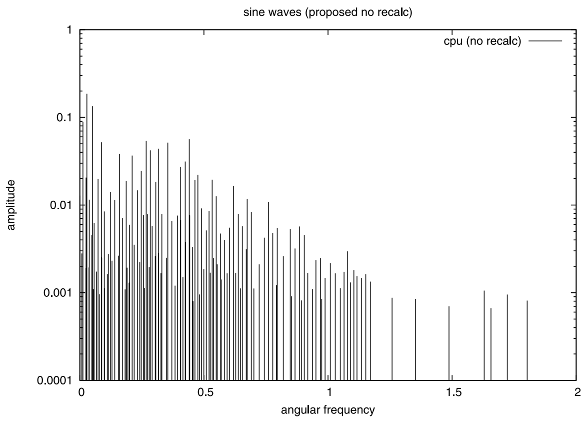


Fig. 8 Extracted sinusoids computed by the proposed no recalculation method by the CPU.

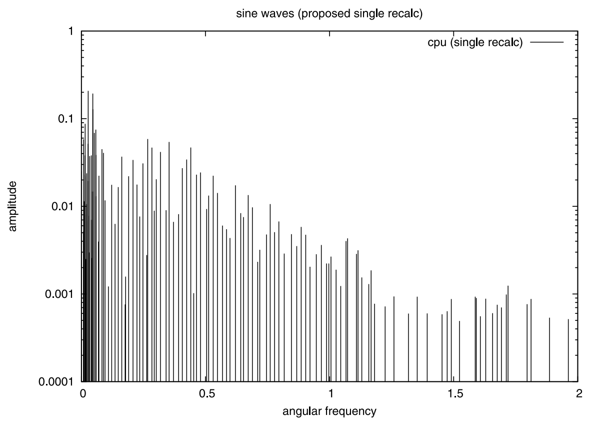


Fig. 9 Extracted sinusoids computed by the proposed single recalculation method by the CPU.

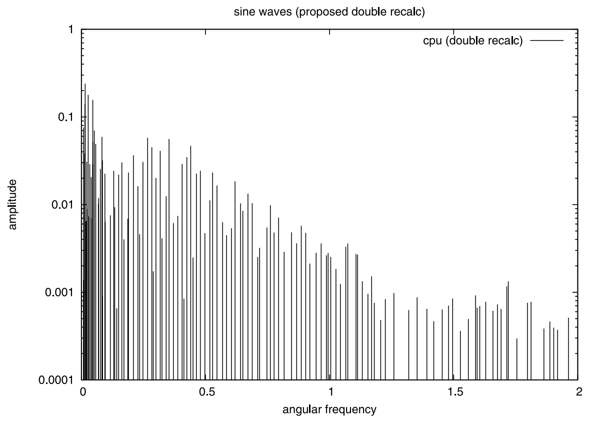


Fig. 10 Extracted sinusoids computed by the proposed double recalculation method by the CPU.

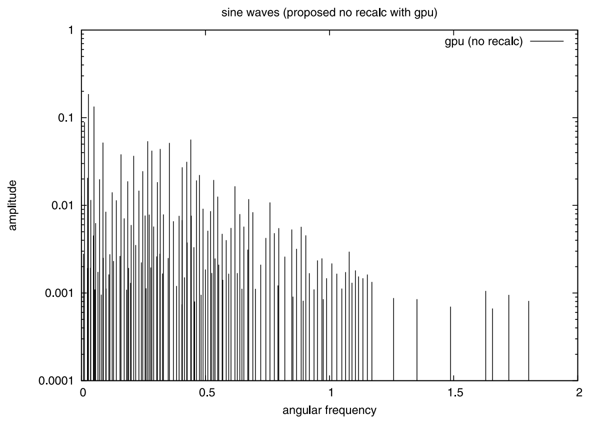


Fig. 11 Extracted sinusoids computed by the proposed no recalculation method by the GPU.

Figure 5 shows the GDL between the original audio signal and the signals resynthesized by Hirata's algorithm and the proposed methods both computed on the CPU. The proposed method without recalculation has 6.86 dB higher GDL compared to Hirata's algorithm. When the single recalculation and the double recalculation methods are adopted, the GDL is 9.45 dB and 11.12 dB higher than Hirata's algorithm, respectively. This means that the proposed methods can resynthesize the signal more accurately than Hirata's algorithm using the same number of sinusoids. In other words, the proposed method can resynthesize the signal with less number of sinusoids than Hirata's algorithm with the same accuracy.

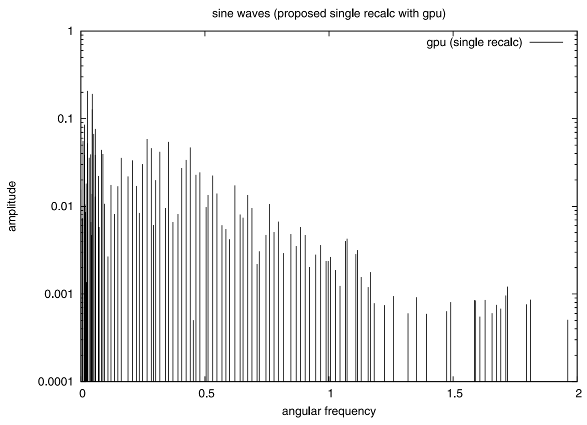


Fig. 12 Extracted sinusoids computed by the proposed single recalculation method by the GPU.

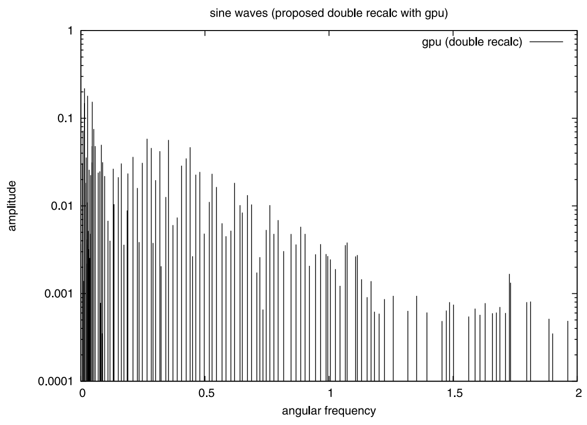


Fig. 13 Extracted sinusoids computed by the proposed double recalculation method by the GPU.

Table 2 GDL (dB) resynthesized 128 sinusoids.

Method	CPU	GPU
Hirata's	−20.65	no data
Proposed Norecalc	−27.50	−27.50
Proposed Recalc Single	−30.29	−31.71
Proposed Recalc Double	−31.77	−31.77

Table 3 Computation time (ms).

Method	Microsoft Visual C++ 9.0	Intel C++ Compiler 10.0	CUDA 2.0
Hirata's	1084862	512042	no data
Proposed No Recalc	165985	86237	2435
Proposed Single Recalc	10603139	5621527	175115
Proposed Double Recalc	10461006	5635848	175258

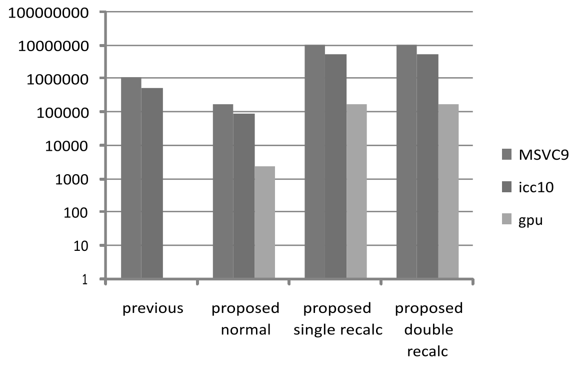


Fig. 14 Computation time.

Figure 6 shows the GDL of the signals parallelly computed by GPU. This results are similar to Fig. 5, which means that the computation accuracy is maintained when GPU is used.

In Figs. 5, 6 and Table 2, the GDLs of 128 sinusoids extracted by single recalculation method between CPU and GPU are different. In the result of GHA, the parameters of after 83-th sinusoid are different between CPU and GPU. This is due to the difference in architectures for floating point computation between CPU and GPU.

Figures 7–10 show the frequencies and the amplitudes of the extracted sinusoids. Compared to Fig. 7, Fig. 8 contains more sinusoids with lower levels in the high frequency band. Since the total numbers of sinusoids are the same, improved accuracy of the proposed method makes it possible to resynthesize the signal with fewer sinusoids, and more sinusoids are extracted in the high frequency range to better approximate the original signal. From the same reason, Figs. 9 and 10 contain more sinusoids than Figs. 7 and 8.

The middle frequency band of Figs. 9 and 10 have less sinusoids compared to Figs. 7 and 8. This means that the redundant sinusoids in the middle frequency range are removed by the recalculation.

Table 3 and Fig. 14 show the computational time of each method and each implementation. The computational times of the proposed method without recalculation are 5-6 times faster than that of Hirata's algorithm simply due to the difference in the number of calculations.

The time needed to compute by GPU is 60–68 times faster than MSVC9, 32–35 times faster than ICC10, and 210–445 times faster than Hirata's algorithm computed on CPU. This verifies the suitability of parallel computation in the proposed method.

There are two main reasons for the improvement in computational time. One is that the inherent parallelism of the proposed algorithm is fully utilized. The other is that GPU has an assembler level instructions for computation of trigonometric functions which are calculated by hardware.

With the recalculation the computational time becomes much longer than both the proposed method without recalculation and Hirata's algorithm. It matches up to the computational complexities of the algorithm. These show that there is a tradeoff between computational time and accuracy. It is thus needed to select the algorithms depending on the purpose.

We also experimented with two other audio signals. One hundred twenty eight sinusoids are extracted with Hirata's method and the proposed method. GDL of the resynthesized signal are compared for each signal.

The first one is a single tone of clarinet. The signal has a fundamental tone, several overtones whose frequencies are integer multiples of the fundamental tone, some noises and flickers. The GDL is −23.03 dB by Hirata's method and is −49.27 dB by the proposed method. This shows that the proposed method can analyze simple audio signals more accurately than Hirata's method.

The second one is a mix of two chirp signals. The frequency of the first chirp signal changes linearly and the frequency of the second chirp signal changes in second order. The GDL by Hirata's method is −12.65 dB and is −68.89 dB by the proposed method. This shows that Hirata's method is not suitable for signals whose frequency changes and the proposed method can analyze them normally.

7. Conclusion

A new method for GHA is proposed, which uses peak estimation of Fourier spectrum and binary search. Samples of windowed Discrete-Time Fourier Transform of each frame is calculated by FFT.

To remove the effect of the windowing two recalculation methods of sinusoidal parameters are also proposed.

The proposed method has high degree of concurrency because frames can be calculated independently, and it can properly be computed in parallel. We implemented the method on GPU.

Using the proposed method, one hundred twenty eight sinusoids are extracted from each frame of 10-second audio signal. Signal is resynthesized and compared with the original audio signal. The proposed method shows 11.2 dB improvement in GDL compared to Hirata's algorithm with maximum of 445 times faster speed. This speed is faster than the playback speed of the audio signal.

References

[1] N. Wiener, "Generalized harmonic analysis," Acta Mathematica, vol.55, pp.117–285, 1930.

[2] Y. Hirata and T. Koike, "Speech band compression using a generalized harmonic analysis," IEICE Technical Report, EA98-3, 1998.

[3] M. Nakazawa and Y. Yamasaki, "Sound coding using 1/12n octave analysis," GITS/GITI Research Bulletin, vol.2002, pp.81–85, July 2003.

[4] R. Takamizawa, K. Katayama, Y. Kanda, and T. Muraoka, "Scratch noise reduction of sp record utilizing generalized harmonic analysis (gha)," IPSJ SIG Technical Reports. SLDM, vol.2004, no.102, pp.1–6, Oct. 2004.

[5] E.B. George, "Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones," J. Audio Eng. Soc., vol.40, no.6, pp.497–515, 1992.

[6] S. Ushiyama, M. Tohyama, M. Iizuka, and Y. Hirata, "Generalized harmonic analysis of non-stationary waveforms," IEICE Technical Report, EA93-103, 1994.

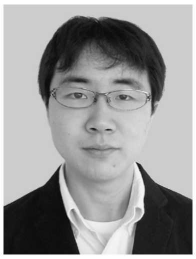
[7] M. Tohyama and T. Koike, "High resolution frequency analysis," Journal Acoust. Soc. Jpn. (E), vol.54, no.8, pp.568–574, 1998.

[8] T. Terada, "Nonstationary waveform analysis and synthesis using generalized harmonic analysis," IEEE TF/TS Symp., pp.429–432, 1994.

[9] T. Muraoka and S. Kiriu, "Reduction of frequency searching processes for generalized harmonic analysis (gha)," IEICE Technical Report, DSP03-1, 2003.

[10] gpgpu.org, "General-purpose computation on gpus (gpgpu)," http://gpgpu.org/

[11] N. Corp. NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 2.0 beta 1, 2008.



Hisayori Noda received the B.E. in Computer Science and M.E. in Communications and Integrated Systems from Tokyo Institute of Technology in 2006 and 2008, respectively. He is now a doctor course student of the Department of Communications and Integrated Systems, Tokyo Institute of Technology. His main research interests are in signal analysis and processing especially for audio applications.

Akinori Nishihara received the B.E., M.E. and Dr.Eng. degrees in electronics from Tokyo Institute of Technology in 1973, 1975 and 1978, respectively. Since 1978 he has been with Tokyo Institute of Technology, where he is now Professor of the Center for Research and Development of Educational Technology. His main research interests are in one- and multi-dimensional signal processing, and its application to educational technology. He has published more than 200 technical papers in international journals and conferences. He served as an Associate Editor of the IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences from 1990 to 1994, and then an Associate Editor of the Transactions of IEICE Part A (in Japanese) from 1994 to 1998. He was an Associate Editor of the IEEE Transactions on Circuits and Systems II from 1996 to 1997 and Editor-in-Chief of Transactions of IEICE Part A (in Japanese) from 1998 to 2000. He has been serving in IEEE Region 10 Executive Committee, as Student Activities Committee Chair (1996–1996), Treasurer (1999–2000), Educational Activities Committee Chair (2001–2004) and Bylaws and Operations Manual Coordinator (2007–2008). He also served as an Executive Committee Member of IEEE Tokyo Section (1995–2004) and IEEE Japan Council (1999–2004). He served as a member of the Board of Governors, IEEE Circuits and Systems Society (2004–2005). He was Chair or the IEICE Technical Group on Circuits and Systems from 1997 to 1998, and since 1998 he has been serving as an Advisor of that Technical Group. He is now serving as Vice President, Service Activities of the IEICE Engineering Sciences Society. He received Best Paper Awards of the IEEE Asia Pacific Conference on Circuits and Systems in 1994 and 2000, a Best Paper Award of the IEICE in 1999, and IEEE Third Millennium Medal in 2000. He also received a 4th LSI IP Design Award in 2002. Prof. Nishihara is a Fellow of IEEE, and a member of EURASIP, European Circuits Society, Association for Advancement of Computing in Education, and Japan Society for Educational Technology.