SINUSOIDAL SYNTHESIS USING A FORCE-BASED ALGORITHM

Ryoho Kobayashi

Faculty of Environment and Information Studies,
Keio University SFC
Kanagawa, Japan
ryoho@sfc.keio.ac.jp

ABSTRACT

In this paper we propose a synthesis method using a force-based algorithm to control frequencies of multiple sine waves. In order to implement this synthesis method, we analyze an existing sound source using a fast Fourier transform (FFT). Spectral peaks which have large magnitudes are regarded as heavy partials and assigned large attractive forces. A few hundred sine waves with stationary amplitudes are placed in a frequency space on which forces generated in the analysis phase are applied. The frequencies of the partials gravitate to the nearest peak of the reference spectrum from the source sound. As more sine waves are combined at the large peaks, the sound synthesized by the partials gradually transforms into the reference spectrum. In order to prevent the frequencies of the partials from gravitating onto localized peaks, each partial is assigned a repulsive force against all others. Through successful control of these attractive and repulsive forces, roughness and speed variation of the synthesis can be achieved. Moreover, by increasing or decreasing the number of partials according to the total amplitude of the source sound, amplitude envelope following is achieved.

1. INTRODUCTION

A force-based (or force-directed) algorithm is commonly known as a graph drawing algorithm [1]. A graph is a common data structure which is constructed from a set of vertices and edges, where the edges connect pairs of vertices [2]. The synthesis method proposed in this paper is inspired by this algorithm, and utilizes the algorithm in order to generate sound.

The motivation of this research is to accomplish a new synthesis method as an application of the sinusoidal partial editing technique [3, 4]. The basic premise of the synthesis method is placing multiple sinusoidal waves which have separate frequencies, and applying a one dimensional force-based algorithm in a frequency domain to control the frequencies of the waves. This method is different from general spectral editors such as spectral SPEAR [5] in that it is not developed for flexible sound editing, but rather for generating characteristic time-varying sounds between noises and recorded sound materials.

The goal of this research is to develop a synthesis method which can generate various sounds from musical tones and noises with a small number of intuitive parameters. In order to achieve this goal, we prepare an existing sound to generate attractive forces and apply them to the force-based algorithm. Strong attractive forces are assigned to large peaks in the spectrum by analyzing the reference sound source using the Fourier transform. A user can vary the similarity of the sound to the reference sound by controlling the forces applied.

All programs presented in this paper are written in Objective-C and C++ and are executed in Mac OSX.

2. STRUCTURES

The structures used to achieve the synthesis method are described in this section.

2.1. Analysis of a reference sound source

The first step is the analysis of a reference sound selected by a user. The reference sound is usually provided as a sound file except for real-time processing, which is presented in section 3.3. Any valid sound source is allowed.

The Short-Time Fourier Transform (STFT) analysis [6] is used for this step, where amplitudes A for each frequency f are detected by

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-2\pi i k n / N} \tag{1}$$

$$A(k) = |X(k)| \tag{2}$$

$$F(k) = \frac{kR}{N} \tag{3}$$

where $x(n)$ consists of N samples of a windowed waveform and R represents the sampling rate.

2.2. Distribution of partials

The synthesis phase for the proposed method begins by generating partials in a specific range of frequencies. The amplitudes of the partials are calculated from the number of partials and are unchangeable. There are two options for determining the number of partials.

1. The user specifies the number of partials and it does not change. The synthesized result does not follow the amplitude of the input reference source.
2. The user specifies the maximum number of partials and the number of active partials is determined in proportion to the amplitude of the reference sound as (4).

$$\nu = \nu_{max} \sum_{k=0}^{N-1} \alpha A(k) \tag{4}$$

where α represents a constant number for scaling the amplitude. The active or inactive partials are randomly chosen. In this step, since the frequencies of the partials are random, an unpitched sound is typically created.

2.3. Attractive force

Attractive forces, which are applied to the partials, are generated from the spectrum detected from the reference sound. A partial is attracted to neighboring frequency components, where the user can specify the number of effective frequency components. The force is inversely proportional to the square of the distance between the target frequency component and the frequency of the partial

$$f_a(P_f(i)) = \sum_{0<|F(k)-P_f(i)|<\tau} \frac{\mathrm{sgn}(F(k) - P_f(i))g_a A(k)}{|F(k) - P_f(i)|^2} \quad (5)$$

where $f_a(P_f(i))$ represents an attractive force for partial $P(i)$ of which the frequency is $P_f(i)$, g_a is a constant value to adjust the strength of the force, and τ corresponds to the range of the effective frequency components.

Figure 1 depicts an example of attractive forces which are applied to a partial. Three peaks of the spectrum are used for calculation in this figure. A large and close peak such as A has a profound effect while a distant peak such as C has little effect. As a result, the effect of peak A is significant, thus this partial shifts to the left side (lower in the frequency domain).

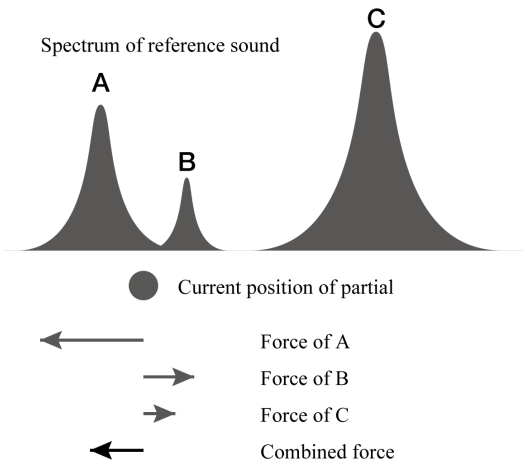

Figure 1: *Attractive forces applied to a partial*

2.4. Repulsive force

To avoid congestion of partials at a small peak in the spectrum, repulsive forces are generated between every pair of partials. The force is inversely proportional to the square of the distance between the partials

$$f_r(P(i)) = \sum_{P_f(j)\neq P_f(i)} \frac{\mathrm{sgn}(P_f(i) - P_f(j))g_r}{|P_f(i) - P_f(j)|^2} \quad (6)$$

where $f_r(P(i))$ represents a repulsive force for partial $P(i)$. By using all pairs of partials for the calculation, partials depart from condensations.

Figure 2 represents repulsive forces between each pair of three partials. Since the partials repel close partials more strongly, partial B is moved up.

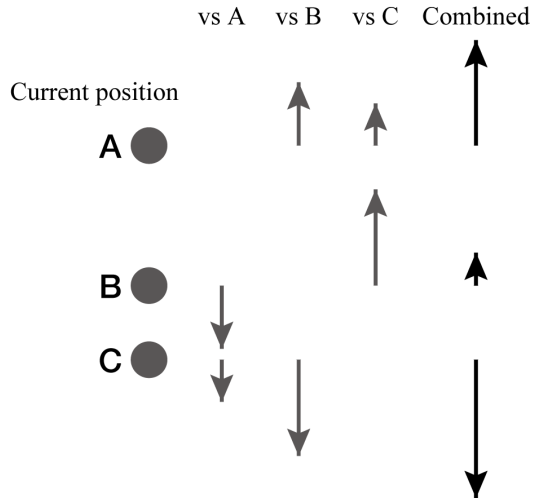

Figure 2: *Repulsive forces applied to three partials*

When some partials have the same frequencies, the repulsive forces between these partials are unable to activate and the attractive forces are always congruent. To separate these partials, two options which user can select are prepared.

1. Random repulsive forces are applied to each partial which is placed at the same frequency with other partials.
2. New frequencies, which is unrelated to current status, are redistributed to the partials.

When some partials have the same frequencies, the repulsive forces between these partials are not activated and the attractive forces are always congruent. In order to separate these partials, there are two options for the user:

1. Random repulsive forces are applied to each partial which is placed at the same frequency with other partials.
2. New frequencies, which are unrelated to the current status, are redistributed to the partials.

Since it is infrequent that multiple partials have exactly the same frequency, the difference in the final synthesized sound qualities is minimal between the aforementioned options.

2.5. Resistance

When the reference sound has a static frequency component, the partials have the risk of periodic vibration around a spectral peak. This is because the attractive forces convert back and forth between potential and kinetic energy. Therefore, the oscillations are inhibited by implementing resistance.

$$f(P(i)) = r(f_a(P(i) + f_r(P(i)) \quad (7)$$

produces total force $f(P(i))$ for partial $P(i)$. r is a resistance value between 0 and 1.

2.6. Synthesis

The forces, which are derived in section 2.5, are applied to partials at every frame by addition of the forces

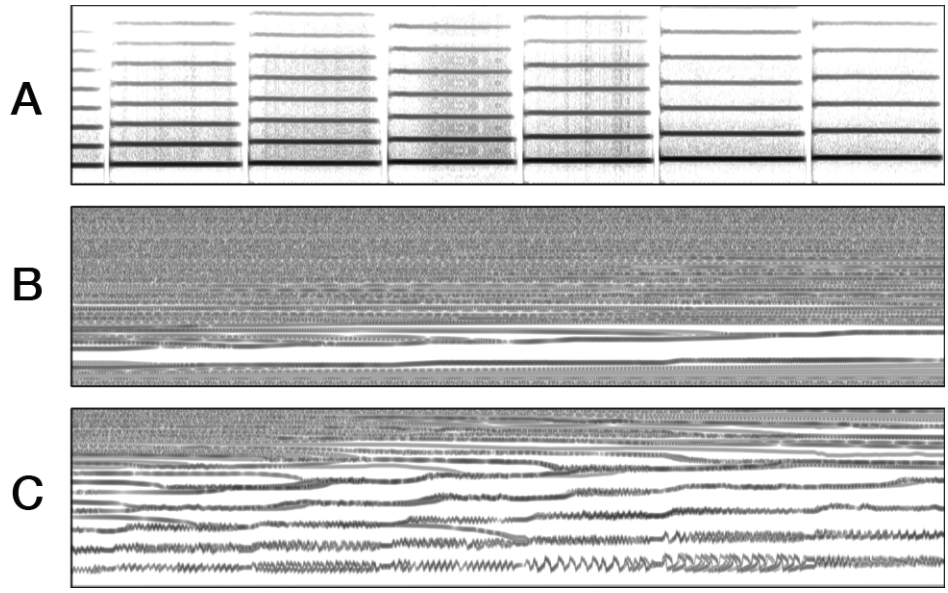

Figure 3: *An example result of the synthesis (the horizontal axis is time and the vertical axis is frequency). "A" is the reference sound (Horn; from 0Hz to 2,000Hz), while "B" and "C" are the synthesized sounds which consist of 1000 partials. The attractive force (ga) for "B" is 0.3, and gr for "C" is 5.*

$$P_f(n,i) = P_f(n-1,i) + f(P(n-1,i)) \tag{8}$$

where n represents the current time frame.

The sound synthesis is accomplished using a common oscillator bank synthesis technique [7, 8] which is realized by

$$y(n) = \sum_{\forall i} A\cos[2\pi P_f(n,i)n + \phi_i] \tag{9}$$

where A represents a constant amplitude for each partial, and an initial phase ϕ_i is randomly distributed.

An example of the synthesis result is depicted in Figure 3. A is generated from a horn sound; B and C are the synthesis results which consist of 1000 partials. The beginning of the synthesized sound starts with an unpitched noise at the left in the figure and the pitched tone is constructed gradually.

It is also possible to use the inverse Fourier transform for both simplified and/or real-time calculation which is proposed in the section 3.3.

3. APPLICATIONS

In this section, we present some applications for this synthesis method.

3.1. Dynamic control of parameters

This synthesis method can generate various sounds by adjusting the parameters. In particular, the coefficients for attractive force g_a, repulsive force g_r, and resistance r are important for controlling the similarity to the reference sound and the quickness of transitions.

It is possible for users to control these parameters by preparing time-varying functions. These functions are written in files which the synthesis program reads in order to synthesize the sound.

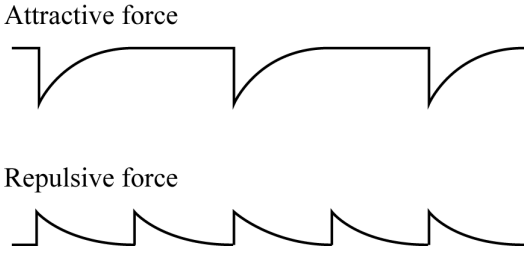

Figure 4: *An example of the force functions used to generate rhythmic sound.*

For instance, periodic changes in timbre are generated by using the functions depicted in Figure 4, where a rhythm structure is created. Control of the resistance value is effective to generate and remove vibrations. Moreover, various effects are realized by applying these forces to the amplitudes of the reference sound without preparing function files.

3.2. Timbre morphing

Timbre morphing is a technique used to create a new sound by combining multiple sounds and transforming the timbres gradually [9]. Sinusoidal modeling, which is used for the synthesis method in this paper, is utilized in some cases of timbre morphing [10, 11]. This synthesis method does not analyze the multidirectional features of the reference sound and it is therefore difficult to generate a high-quality and well-defined result. However, uncomplicated timbre morphing is accomplished by using this synthesis method.

A sound file which consists of several segments from separate sound sources is required to accomplish timbre morphing, using

the file as a reference. Each particle for this synthesis method varies continuously; therefore, the generated timbre transforms gradually at the transition of the reference sounds.

By decreasing the attractive force and increasing the repulsive force at the transition, the synthesized sound converges to the desired sound, and a smooth morphing result is achieved.

3.3. Real-time processing

The synthesis method proposed in this paper generates a wide variety of sounds and controls them through a small number of parameters; therefore, this method has the potential for realizing a novel and intuitive user interface for generating sounds. At this time, it is difficult for off-the-shelf personal computers to generate high-quality results in real time due to the large calculations required.

By implementing the ideas below, real-time processing is accomplished in computationally limited environments.

1. Decreasing the frame rate(e.g. 30fps)
2. Using a Graphics Processing Unit (GPU)
3. Decreasing the number of partials (e.g. 200)
4. Using the IFFT for synthesis

GPUs are able to realize fast parallel processing; hence it is effective to calculate the frequencies of many partials using them. In this research, the GLSL (OpenGL Shading Language) is used [12, 13].

The IFFT (inverse fast Fourier transform) is a common synthesis method which is used for decreasing the calculation time. In addition to this, the calculation volume is reduced by decreasing the FFT size and frame rate. However, deteriorations in resolution in both the time and frequency domains are observed, meaning the generated sound also has a loss in quality.

4. CONCLUSION

In this paper, structures and applications of a new synthesis method, which is constructed using sinusoidal editing and a force-based algorithm, were proposed. Although this method is a complicated tool at this time, examples of productive features are indicated.

The following two points are considered important for the future prospects of this research:

1. Development of real-time processing and an intuitive and interactive user interface.
2. Adoption of advanced physical laws.

As mentioned in section 3, this method has the potential to realize various sound synthesis. However, it is difficult to utilize this method for musical performances and real-time and/or interactive compositions due to the large volume of calculations required.

This method also has the possibility to involve various physical laws, for instance, recent research results of fluid mechanics indicate large potentials for controlling a vast amount of particles and streams. The synthesis method presented in this paper has already accomplished the creation of a wide variety of sounds using a simple force-based model. It is proposed that more powerful, flexible, and intuitive synthesis methods can be realized by utilizing advanced models.

5. REFERENCES

[1] Thomas M. J. Fruchterman and Edward M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

[2] John Adrian Bondy and Uppaluri Siva Ramachanda Murty, *Graph Theory with Applications*, The Macmillan Press Ltd., London, 1976.

[3] Robert McAulay and Thomas F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[4] Kelly Fitz and Lippold Haken, "Sinusoidal modeling and manipulation using lemur," *Computer Music Journal*, vol. 20, no. 4, pp. 44–59, 1996.

[5] Michael Klingbeil, "Software for spectral analysis, editing, and synthesis," in *Proceedings of the International Computer Music Conference*, 2005, pp. 107–110.

[6] Jont B. Allen, "Short term spectral analysis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics, Speech, and Processing*, vol. 25, no. 3, pp. 235–238, 1977.

[7] G. DiGiugno, "A 256 digital oscillator bank," in *Proceedings of the International Computer Music Conference*, MIT, Cambridge, 1976, pp. 188–91.

[8] F. Richard Moore, *Elements of Computer Music*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1990.

[9] John M. Grey, *An Exploration of Musical Timbre*, Ph.D. thesis, Stanford University, 1975.

[10] Lippold Haken Edwin Tellman and Bryan Holloway, "Timbre morphing of sounds with unequal numbers of features," *Journal of the Audio Engineering Society*, vol. 43, no. 9, pp. 678–689, 1995.

[11] Kelly Fitz Lippold Haken and Paul Christensen, *Analysis, Synthesis, and Perception of Musical Sounds*, chapter Beyond traditional sampling synthesis: Real-time timbre morphing using additive synthesis, pp. 122–14, James W. Beauchamp Eds. Springer, New York, 2007.

[12] John D. Owens et al., "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[13] Ricardo Marroquim and André Maximo, "Introduction to gpu programming with glsl," in *Tutorials of the XXII Brazilian Symposium on Computer Graphics and Image Processing*, Rio de Janeiro, Brazil, 2009, pp. 3–16.

6. APPENDIX: SOUND EXAMPLES

Sound examples are available online at the following address.

http://www.ryoho.com/software/sinfba/