Squawk: A Graphical Software for Spectral Audio Processing

Ryoho Kobayashi

Graduate School of Media and Governance, Keio University
ryoho@sfc.keio.ac.jp

Abstract

This paper describes the design and development of new graphical software for spectral audio processing. There are three phases to accomplish the audio processing. First, spectrograms are generated from an input audio signal using Short-Time Fourier Transform (STFT) analysis, and the sonogram image is continually shifting from right to left. Secondly, the sonogram is transformed by placing prepared effect objects such as panning, compressor and delay, on the streaming spectrogram. These effect objects provide various transformations to specific range of frequencies, then, by combining them, powerful effects will be produced. Thirdly, a transformed audio signal is generated from the edited sonogram by using overlap-add resynthesis or oscillator bank resynthesis technique.

1 Introduction

Spectral audio processing is an effective technique for synthesizing new sounds from existing materials, and numerous applications have been published over the past few decades. There are, however, a considerable number of parameters for controlling these processors, therefore, it is difficult to handle spectral data completely.

For reasons mentioned above, it is obvious that an ingenious user interface is required. Various applications such as MetaSynth, NI-Spektral Delay, AudioSculpt (Serra 1997), and Lemur (Fitz, Haken, and Holloway 1995) have actually demonstrated the power and popularity of graphical user interfaces for spectral audio processing.

The new application software named Squawk this paper presents currently run on Mac OSX, and provides a simple graphical user interface based on spectrographic transformation techniques (Roads 2001).

The purposes of the design and development of this software are to produce powerful, flexible and various spectral transformations with intuitive operationality. To accomplish these purposes, some new ideas adopting Spectral Stream, and Effect Object, are implemented for the graphical user interface.

2 Analysis

The analysis in Squawk adheres to the Short-Time Fourier Transform (STFT) technique (Allen 1977, Moore 1990, Kobayashi 2003). And, the results of the STFT analysis are utilized to generate spectrograms.

2.1 STFT analysis

The first step for analysis is to calculate magnitude and phase spectra from an audio signal using the STFT analysis. As a preparation for spectrum analysis, the STFT imposes a window upon the input signal. This windowing process breaks the input signal into a series of segments that are shaped in amplitude by the chosen window function. By adopting DFT techniques for each windowed and segmented input signal, both the magnitude and phase spectrum are provided. For further details of the STFT techniques, see Allen (1977) and Moore (1990).

In Squawk, the user can set general parameters for STFT (FFT size, window size, hop size and window type).

2.2 Generating Spectrograms

A spectrogram is a well-known spectrum display technique. It represents an audio signal as a two-dimentional display of time versus frequency.

In Squawk, four spectrograms are used. These describe magnitude spectrograms and phase spectrograms for each stereo channel. In case of where a mono sound source is provided, the same spectrogram is adopted for each stereo channel in the analysis stage.

The phase spectrograms, which are containing phase values for each time and frequency, are not displayed in editor field, and they are used only for resynthesis.

For the magnitude spectrograms, the magnitudes for each time and frequency for left channel are converted to the brightness of blue components, and for right channel, they are converted to the brightness of yellow components. By adding these two spectrograms, the magnitude spectrograms for each stereo channel are displayed on a single image.

The user can select the types of scale for magnitudes and frequencies, whether linear or logarithmic.

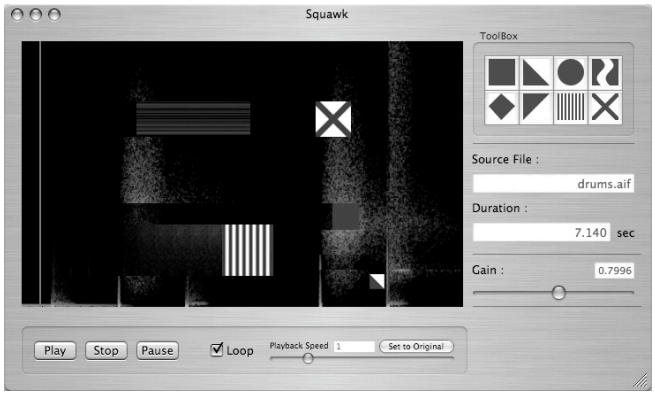

Figure 1. Squawk user interface. Cutoff Object, Shifting Object, Delay Object, and Snapshot Object are placed in a sonogram field.

3 User Interface

In this section, design of user interface for this software is illustrated. The Squawk user interface offers intuitive interaction for spectral transformation by adopting original ideas such as Spectral Stream and Effect Object.

3.1 Basic Parameters

Squawk has some basic parameters for analysis, editing, and resynthesis. General STFT parameters, the range of frequencies and amplitudes, and the type of scale (linear or logarithmic) for amplitudes and frequencies for display, are set on the configuration window. Repetition, speed, and gain for playback can be adjusted on the main window.

3.2 Spectral Stream

The spectrograms provided by STFT analysis are displayed on the main window, and shifted continually from right to left. The speed of shifting corresponds to playback speed. By changing this value, time-stretching effect is briefly provided.

3.3 Spectrographic Transformation

The spectrographic transformations are produced by Effect Objects. Every object is rectangular shaped, and has a mark corresponding to the effect. When frequency components, represented in spectrogram, touch the right side of an object, they are regarded as an input for the processor, and then, the processed results will be output from the left side of the object. This operation denotes that the height of a object corresponds to the range of frequencies for a processor (Figure 2).

The detailed functions for each object are presented in the next section.

3.4 Synthesis Line

In Squawk, time transition is represented as the movements of spectrograms. A fixed vertical line is displayed on the main window to specify the position for resynthesizing. The spectrum approaching to this line will be resynthesized and converted to a time-domain audio signal (Figure 2).

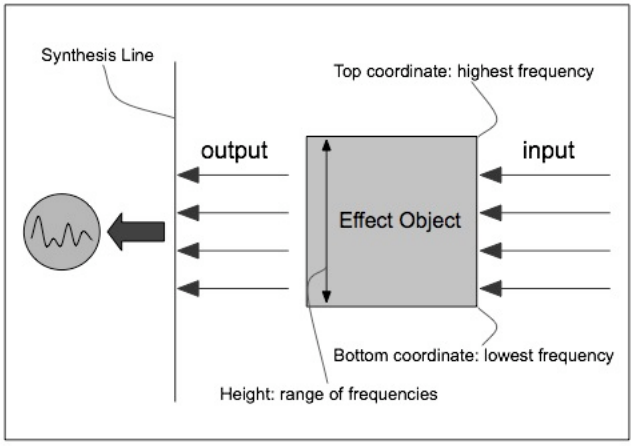

Figure 2. Behaviors of "Effect Object" and "Synthesis Line". The vertical axis depicts frequency, and the horizontal axis depicts time.

4 Parameters for Effect Objects

Squawk has eight effect objects for editing spectra. In this section, parameters and functions for each effect object are described.

Every object implemented for Squawk has three common parameters that are described in the following list.

Origin	Location of the origin of effect object. The X-axis position corresponds to time, and the Y-axis position corresponds to frequency.
Size	Width and Height of effect object. The height corresponds to bandwidth of frequency.
Dry / Wet	Adjust the balance of original and effected sound

4.1 Cutoff Object

Cutoff Object deletes frequency components within a specific range of frequencies that is provided by the location and size of the object. This object has no additional parameters.

4.2 Panning Object

Panning Object provides an auto panning effect, which creates a time-varying movement between left and right for

stereo. In the Squawk user interface, this object converts the color of sonogram.

Center	Shift center position for modulation between left (-100) and right (100). Default value is 0.
Amount	Amplitude of modulation.
Frequency	Frequency of modulation. This value is set in Hz or BPM.
Mod-Type	Waveform for modulator, sine, sawtooth, square, triangle, or random.

4.3 Shifting Object

Shifting Object creates modulated pitch shifting, by shifting up or down a specified range of frequencies.

Base	Shifting amount in Hz. Positive value means shift-up and negative value means shift-down.
Amount	Amplitude of modulation.
Frequency	Frequency of modulation. This value is set in Hz or BPM.
Mod-Type	Waveform for modulator, sine, sawtooth, square, triangle, or random.

4.4 Stretching Object

Stretching Object creates frequency stretching. This operation means converting a range of frequencies.

Center	Center frequency for stretching.
Ratio	Stretching ratio between 0 and 100. The value 0 makes single partial tone (one pixel in Squawk user interface).

4.5 Compressor Object

Compressor Object alters the dynamic envelope.

Threshold	Level in dB past which compression will begin to take effect.
Ratio	Amount of gain reduction.
Attack	How fast the gain is reduced in response to an increase in input signal level above the threshold.
Release	How fast the gain is restored to normal levels after the input signal falls below the threshold.

4.6 Delay Object

Delay Object offers stereo delay effects. The parameters Delay Time and Feedback are prepared for each left and right channel.

Delay Time	Specify the delay time in milliseconds or BPM.
Feedback	Multiplier for feedback delay.
Stereo Link	Link effect for each left and right channel.

4.7 Filter Object

Filter Object applies a filter in frequency domain within a specific range of frequencies. By drawing the shape of filter, a flexible effect is produced (Figure 3).

Shape	Specify the delay time in milliseconds or BPM.

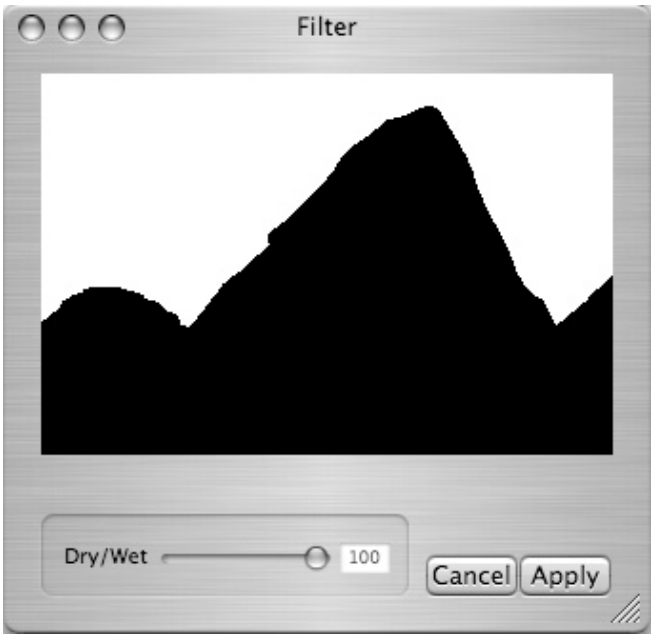

Figure 3. Parameter setting window for filter object

4.8 Snapshot Object

Snapshot Object varies the intervals of transition in time domain. By taking a sound frame, and by holding the frame stationary until a next cue is given, a rhythmic transition is produced.

Interval	Interval between cues in milliseconds or BPM.

5 Resynthesis

The user has two options, overlap-add resynthesis or oscillator bank resynthesis technique, for generating an audio signal from a transformed spectrogram.

5.1 Overlap-Add Resynthesis

By adopting Inverse Discrete Fourier Transform (IDFT) to each frame, each windowed and segmented signal is reconstructed from the spectrum components. The IDFT takes each magnitude and phase component from spectrograms, and generates a corresponding time-domain signal. Then, by overlapping and adding these resynthesized segments, this method provides an audio signal.

5.2 Oscillator Bank Resynthesis

Oscillator bank converts the analysis data, which are obtained from magnitude and phase spectrograms, into the sets of amplitude and frequency envelopes for controlling oscillators. This method is generally powerful for spectral transformations than the overlap-add method, and provides high-quality results in most situations for spectrographic editing. This method, however, requires high computational cost. For reducing the cost, it is effective to narrow the range of frequencies for resynthesis. Squawk can specify the displaying range of frequencies.

6 Conclusion

Squawk offers powerful tools for spectral transformation with easy-to-use graphical user interface. Squawk has, however, a few limitations.

Squawk currently has only eight processors. There are numerous processing techniques can be added. To add these various processors, Squawk should eventually implement an original plug-in format that is specialized in spectrographic transformations, and publish SDK for the plug-in.

As another limitation, there are difficulties of real-time manipulation. In the current version of Squawk, the parameters for processors cannot be altered. MIDI or OpenSound Control (Wright, Freed, and Momei 2003) will facilitate to solve this problem.

In addition to these, Square will be a performance tool by implement a new composition capability. It is considered that Squawk has compatibility with agent-based composition system (Uozumi 2005).

These ideas will be implemented in future versions.

The demonstration movies, sample sounds and detailed screenshots of current version of Squawk are available at the following web site.

http://web.sfc.keio.ac.jp/~ryoho/squawk/

References

Allen, J. B. 1977. Short Term Spectral Analysis, and Modification by Discrete Fourier Transform. *IEEE Transactions on Acoustics, Speech, and Processing* 25(3), pp. 235-238.

Dolson, M. 1986. The Phase Vocoder – A Tutorial. *Computer Music Journal* 10(4), pp. 14-27.

Fitz, K., L. Haken, and B. Holloway. 1995. Lemur – A Tool for Timbre Manipulation. In *Proceedings of the International Computer Music Conference*, pp. 158-161. Banff, Canada: International Computer Music Association.

Kobayashi, R. 2003. Sound Clustering Synthesis Using Spectral Data. In *Proceedings of the International Computer Music Conference*, pp. 239-241. Singapore: International Computer Music Association.

Moore, F. R. 1990. *Elements of Computer Music*. Englewood Cliffs, New Jersey: Prentice Hall.

Penrose, C. 1999. Extending Musical Mixing: Adaptive Composite Signal Processing. In *Proceedings of the International Computer Music Conference*, pp. 508-511. Beijing, China: International Computer Music Association.

Roads, C. 2001. *Microsound*. Cambridge, Massachusetts: MIT Press.

Serra, M. -H. 1997. Introducing the phase vocoder. In *Musical Signal Processing*. Lisse: Swets and Zeitliinger. pp. 31–90.

Uozumi, Y. 2005. Gismo: An Application for Agent-Based Composition. In *Proceedings of the International Computer Music Conference*, pp. 817-820. Barcelona, Spain: International Computer Music Association.

Wright, M., A. Freed, and A. Momei. 2003. OpenSound Control: State of the Art 2003. In *Proceedings of International Conference on New Interfaces for Musical Expression*, pp. 153-159. Montreal, Canada.