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

1 Introduction
Since spectral audio processing is an effective technique for synthesizing new sounds from existing materials, numerous applications have been published over the past few decades. However, there are a considerable number of parameters for controlling these processors. Therefore, it is not easy to handle spectral data completely.

Accordingly, it is obvious that the new ingenious user interface is required. Various applications such as MetaSynth, NI-Spectral 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.

This paper presents the new application named “Squawk”. It is the new software that currently runs on Mac OSX, and provides a simple graphical user interface based on spectrographic trans-formation techniques (Roads 2001).

The purposes of the design and development of this software are to produce powerful, flexible and various spectral transformations with intuitive operability. 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 by using the STFT analysis. As a preparation for spectrum analysis, the STFT imposes time windows 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 applying the discrete Fourier transform to each windowed / 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 method. It represents an audio signal as a two-dimensional display of time versus frequency.

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

Phase spectrograms that are containing phase values for each time and frequency are not displayed in the 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. 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.
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. They are general STFT parameters, the range of frequencies and amplitudes, and the type of scale (linear or logarithmic) for amplitudes and frequencies for display. These 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 continuously from right to left. The speed of shifting corresponds to the playback speed. By changing this speed value, time-stretching effect is provided immediately.

3.3 Spectrographic Transformation

The spectrographic transformations are implemented by using the “effect objects”. Every object is rectangular shaped, and has the different pattern corresponding to the particular effect. When frequency components that are represented in spectrogram touch the right side of an object, they are regarded as an input for the processor. Then, the processed results will be output from the left side of the object. This operation denotes that the height of an 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).

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Location of the origin of effect object. The X-axis position corresponds to time, and the Y-axis position corresponds to frequency.</td>
</tr>
<tr>
<td>Size</td>
<td>Width and Height of effect object. The height corresponds to bandwidth of frequency.</td>
</tr>
<tr>
<td>Dry / Wet</td>
<td>Adjust the balance of original and effected sound</td>
</tr>
</tbody>
</table>

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 spectrogram.

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

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

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

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

<table>
<thead>
<tr>
<th>Center</th>
<th>Center frequency for stretching.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio</td>
<td>Stretching ratio between 0 and 100. The value 0 makes single partial tone (one pixel in Squawk user interface).</td>
</tr>
</tbody>
</table>

4.5 Compressor Object
Compressor Object alters the dynamic envelope.

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

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

<table>
<thead>
<tr>
<th>Delay Time</th>
<th>Specify the delay time in milliseconds or BPM.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback</td>
<td>Multiplier for feedback delay.</td>
</tr>
<tr>
<td>Stereo Link</td>
<td>Link effect for each left and right channel.</td>
</tr>
</tbody>
</table>

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        | Frequency envelope for filter. X-axis represents frequency, and Y-axis represents amplitude multiplier for frequency components. |

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.                                              |

Figure 3. Parameter setting window for filter object
5 Resynthesis

The user has two options to generate an audio signal from the transformed spectrogram. They are overlap-add resynthesis or oscillator bank resynthesis technique.

5.1 Overlap-Add Resynthesis

By adopting the inverse discrete Fourier transform to each frame, each windowed / segmented signal is reconstructed from the spectrum components. The inverse discrete Fourier transform 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 is 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 compared to the overlap-add method, and mostly provides high-quality results in a situation for spectrographic editing. In Squawk, the user can select the waveform for resynthesis. Oscillator bank resynthesis; 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. However, Squawk still has a few limitations. Squawk currently has only eight processors. There are numerous processing techniques that can be added to Squawk. To add these various processors, Squawk should eventually implement an original plug-in format that is specialized in spectrographic transformations, and should publish SDK for the plug-in.

As another limitation, there are some 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) would help to solve this problem.

In addition to these, Squawk can be a performance tool by implementing 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.

References


