Synthesis of Audio Signals

BE2M31SYN

Semester Project

Project done by:

Juan Lanagrán Jiménez

Date: 31/12/2024

Content of the Project

1. Introduction:

The semester project focuses on implementing custom sound synthesis in MATLAB. The composition may include various sounds, ranging from fully synthetic instruments (e.g., Theremin, Hammond organ, FM synthesizer tones) to everyday sounds. To enhance the quality and realism of the composition, various audio effects such as reverb, echo, chorus, and stereo have been applied.

I have created a composition of the piece *"The Typewriter"* by Leroy Anderson.
In this report, I will discuss the workflow, the code used to perform the audio synthesis, the instruments involved, the techniques applied, the audio effects implemented, and provide explanatory graphs illustrating the waveform and spectrogram of the synthesized instruments.

2. Síntesis de audio “Typewriter”:

We will now discuss the audio synthesis conducted for this semester project, which focuses on recreating the piece "The Typewriter" by Leroy Anderson.
For the synthesis of this composition, we utilized the toolbox provided in the course (MIDI_Toolbox_2.3.zip). This toolbox contains the main functions used in the process, including Main.m, synthchallenge.m, and synth.m, as well as internal functions for working with MIDI files.

The procedure followed is as follows: we begin by selecting the MIDI file to be used and execute the main function, which is essentially a call to synthchallenge.m. Upon executing synthchallenge.m, the MIDI instruments and percussion used in "The Typewriter" are displayed. In this case, the instruments identified are as follows:

```
=======================
Channel 1: instrument no. 1 (718 notes)
Channel 2: instrument no. 3 (499 notes)
Channel 3: instrument no. 10 (50 notes)
Channel 10: instrument no. 15 (577 notes)
      percussion no. 62 (577 notes)
=======================
```

Once the various instruments and percussion have been identified, the synth.m function was modified to incorporate the synthesis of each instrument and percussion element. This was achieved by employing the diverse methods introduced during the course.

Following the code modifications to support the synthesis of this composition, a detailed discussion of the implemented code and the results obtained is presented. Furthermore, an alternative version of the code, named synth_Without_effects.m, was developed. This version allows for testing the synthesis process without the application of audio effects. By comparing the audio output of both versions, the significant impact of the applied enhancements can be clearly observed.

Overview of the Synthesis Code

The MATLAB function synth is the core of the project, performing audio synthesis based on specified parameters such as frequency (freq), duration (dur), amplitude (amp), sampling rate (Fs), synthesis type (synthtype), and channel. The code is structured to handle a variety of synthesis methods for both non-percussion and percussion instruments, as well as apply effects like reverb and echo.

```
function y=synth(freq,dur,amp,Fs,synthtype,channel,note)
% y=synth(freq,dur,amp,Fs,type)
% This function performs audio synthesis based on the specified parameters.
% Added synthesis methods: FM, AM, Karplus-Strong, granular synthesis, and
filtered noise.
% Added audio effects: reverb and echo.
if nargin==5
   channel=1; % Default channel
end
N = floor(dur*Fs); % Number of samples
t = (0:N-1)./Fs;  % Time vector
y = zeros(1, N);  % Initialize output
%% Instrument synthesis based on MIDI type
if channel ~= 10 % Non-percussion instruments
   if synthtype == 1 % Instrument 1: Frequency Modulation (FM)
       mod_freq = freq * 2; % Modulator frequency
       mod_index = 5; % Modulation index
       y = amp .* sin(2 * pi * freq * t + mod_index * sin(2 * pi * mod_freq *
t));
   elseif synthtype == 3 % Instrument 3: Karplus-Strong (string synthesis)
       D = max(2, floor(Fs / freq)); % Ensure delay line length is at least 2
       x = 2 * rand(1, D) - 1; % Initial excitation: white noise
       y(1:D) = x; % Fill the first D samples
       for n = D+1:N
           y(n) = 0.994 * (y(n-D) + y(n-D+1)) / 2; % Feedback with decay
       end
       y = amp * y; % Scale by amplitude
   elseif synthtype == 15 % Instrument 15: Amplitude Modulation (AM)
       mod_freq = freq / 2; % Modulator frequency
       y = amp .* (1 + 0.5 * sin(2 * pi * mod_freq * t)) .* sin(2 * pi * freq
* t);
   else % Default sine wave
       tau = 0.1; % Decay constant
       y = amp .* exp(-t / tau) .* sin(2 * pi * freq * t);
   end
else % Percussion instruments (channel 10)
   if note == 62 % Percussion 62: Filtered noise with distortion
       noise = randn(size(t)); % White noise
       filtered_noise = filter([1, -0.9], 1, noise); % Simple low-pass filter
       y = amp .* tanh(10 * filtered_noise); % Apply distortion
```

```
   else % Default percussion: White noise
       tau = 0.01; % Decay constant
       y = amp .* exp(-t / tau) .* randn(size(t));
   end
end
% Apply audio effects
% Reverb effect
reverb_delay = round(0.2 * Fs); % 200 ms delay
decay = 0.5;
y_reverb = filter([1, zeros(1, reverb_delay - 1), decay], 1, y);
% Echo effect
echo_delay = round(0.4 * Fs); % 400 ms delay
decay = 0.3;
y_echo = filter([1, zeros(1, echo_delay - 1), decay], 1, y_reverb);
% Normalize output to prevent clipping
y = y_echo / max(abs(y_echo(:)));
end
```

Synthesis Techniques Implemented

1. **Frequency Modulation Synthesis (FM)**

   FM synthesis is used to create complex, harmonic-rich timbres by modulating a carrier signal with a modulator signal. In this implementation:
   - The modulator frequency is set to twice the fundamental frequency (freq).
   - The modulation index is used to control the amount of frequency deviation.
     The FM synthesis produces dynamic and rich tones suitable for instruments requiring harmonic complexity, such as bells or metallic sounds.

2. **Karplus-Strong Algorithm**

   The Karplus-Strong method is a form of physical modeling synthesis often used for string-like sounds. The process involves:
   - Initializing the signal with a burst of white noise.
   - Applying feedback and decay using a delay line to simulate the vibration of a plucked string.
     This method is particularly effective for simulating guitar or stringed instruments.

3. **Amplitude Modulation (AM)**

   AM synthesis modulates the amplitude of a carrier signal using a low-frequency oscillator (LFO).
   - The modulation frequency is set to half the fundamental frequency.
   - The resulting signal combines the carrier and the modulation to produce a tremolo-like effect, suitable for instruments with a fluctuating amplitude.

4. **Filtered Noise**

   For percussion instruments, filtered noise is used to generate realistic drum or other percussive sounds.
   - White noise is passed through a low-pass filter to shape the sound.
   - Distortion is applied to create a more dynamic percussive effect.

5. **Default Sine Wave**

   If no specific synthesis type is chosen, the function defaults to generating a simple decaying sine wave. This basic tone is used for general-purpose sound synthesis.

Audio Effects Applied

1. **Reverb**

   To simulate the natural reverberation of a performance space, the code applies a reverb effect by introducing a delayed and decayed version of the signal.
   - A delay of 200 ms is used to simulate early reflections.
   - The decay factor is set to 0.5 to mimic the gradual attenuation of sound in a room.

2. **Echo**

   The echo effect adds repeated, decaying copies of the signal to create depth and a sense of space.
   - A 400 ms delay is used for the echo.
   - The decay factor of 0.3 ensures the echoes fade progressively.

These effects enhance the realism of the synthesized audio, giving the impression of an acoustic performance environment.

Application to "The Typewriter"

For the synthesis of "The Typewriter," the following techniques were used to recreate the distinct timbres of the instruments in the piece:

1. Violin
    - Technique: Filtered sawtooth signal with vibrato.
    - This method generates the bowed string effect, and vibrato adds a natural fluctuation in pitch.
2. Flute
    - Technique: Wavetable synthesis with linear interpolation.
    - A pre-recorded sample is used as the basis, and the pitch is modified to produce melodic notes.
3. Glockenspiel
    - Technique: FM synthesis combined with echoing.
    - This creates a bright, ringing tone characteristic of the instrument.
4. Typewriter Keyboard
    - Technique: Linear Predictive Coding (LPC) and filtered noise.
    - White noise and randomness are added to recreate the percussive clicks of the typewriter keys.

To perform an analysis of the instruments and their sound characteristics, I developed a complementary script to visualize both the spectrogram and waveform. The script, named tech.m, is as follows:

```
% Parámetros generales
Fs = 48000;          % Frecuencia de muestreo
dur = 2;             % Duración en segundos
amp = 0.8;           % Amplitud
synth_types = [1, 3, 15, 0]; % Tipos de síntesis (FM, Karplus-Strong, AM,
Percusión)
channel = [1, 1, 1, 10]; % Canales (1 para instrumentos melódicos, 10 para
percusión)
notes = [69, 72, 76, 62]; % Notas MIDI (violín: A4, flauta: C5, glockenspiel:
E5, typewriter keyboard: percusión)
% Títulos de los instrumentos
instrument_names = {'Violin (FM)', 'Flute (Karplus-Strong)', 'Glockenspiel
(AM)', 'Typewriter Keyboard (Percussion)'};
% Crear figura para visualización
figure;
for i = 1:length(synth_types)
   % Generar sonido
   freq = 440 * 2^((notes(i) - 69) / 12); % Calcular frecuencia a partir de
la nota MIDI (excepto percusión)
   if channel(i) == 10 % Si es canal de percusión
       freq = 0; % Sin frecuencia específica
   end
```

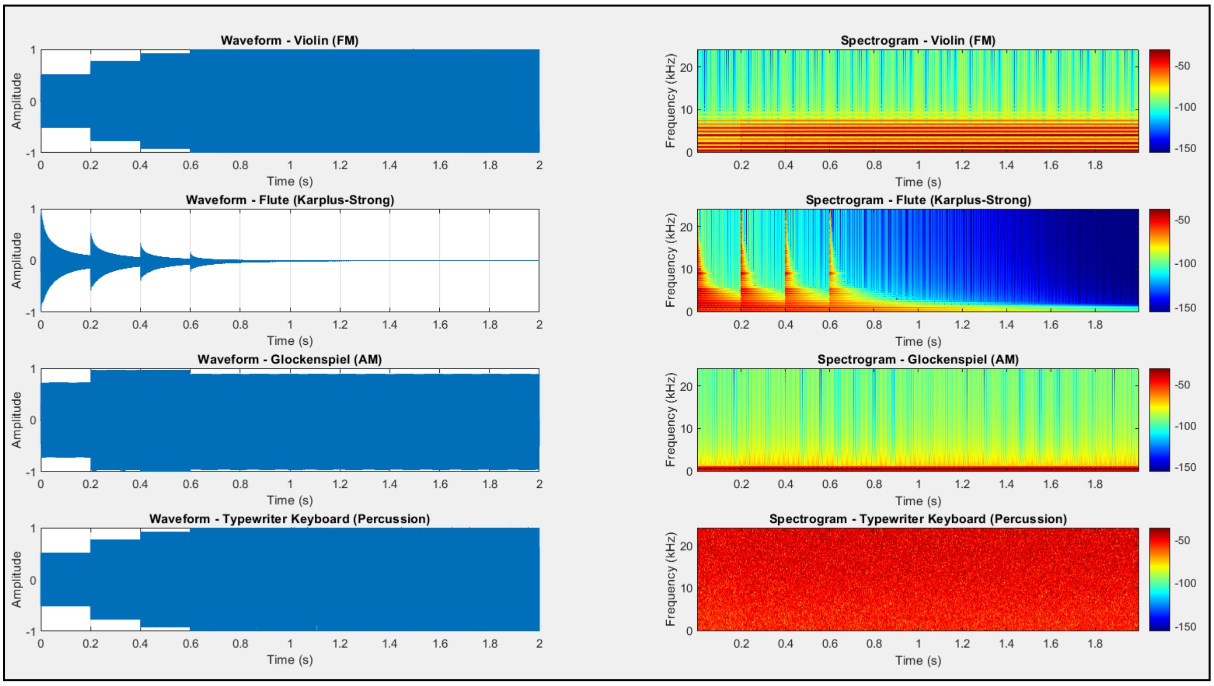
```
    y = synth(freq, dur, amp, Fs, synth_types(i), channel(i), notes(i));

    % Calcular vector de tiempo
    t = linspace(0, dur, length(y));

    % Plot waveform
    subplot(length(synth_types), 2, 2*i-1);
    plot(t, y);
    title(['Waveform - ', instrument_names{i}]);
    xlabel('Time (s)');
    ylabel('Amplitude');
    grid on;

    % Plot spectrogram
    subplot(length(synth_types), 2, 2*i);
    spectrogram(y, 256, [], [], Fs, 'yaxis');
    title(['Spectrogram - ', instrument_names{i}]);
    colormap jet;
    colorbar;
end
disp('Análisis y visualización completa.');
```

The results obtained from running this script are as follows:



Explanation of the Image and Code

Waveform and Spectrogram Analysis

1. **Overview of the Plots:**
   - Waveform: The left column shows the time-domain representation of the audio signals. The amplitude is plotted as a function of time.
   - Spectrogram: The right column visualizes the frequency content over time. The color intensity indicates the amplitude of the frequency components (warmer colors represent higher amplitudes).

2. **Instrument-Specific Observations:**
   - **Violin (FM):**
     - The waveform is a dense sinusoidal signal, typical of Frequency Modulation synthesis.
     - The spectrogram shows harmonics uniformly spaced, representing the rich tonal quality of FM synthesis.
   - **Flute (Karplus-Strong):**
     - The waveform demonstrates a decaying envelope, consistent with string synthesis mimicking the flute's behavior.
     - The spectrogram highlights a few strong harmonics at the start, which fade as the signal decays.
   - **Glockenspiel (AM):**
     - The waveform exhibits an amplitude modulation pattern, with rapid oscillations.
     - The spectrogram reveals periodic frequency bands, characteristic of AM synthesis.
   - **Typewriter Keyboard (Filtered Noise with Percussion):**
     - The waveform contains short bursts resembling random noise with rapid decay.
     - The spectrogram is dense with high-frequency components, representing filtered noise typical of percussion instruments.

3. Conclusion:

This project successfully demonstrates the synthesis of audio signals using various advanced techniques implemented in MATLAB. The recreation of Leroy Anderson's "The Typewriter" was achieved by employing methods such as Frequency Modulation (FM), Karplus-Strong synthesis, Amplitude Modulation (AM), and filtered noise for both melodic and percussive elements. The integration of effects like reverb and echo enhanced the realism of the synthesized sounds, simulating a natural acoustic environment.

Each instrument in the composition was carefully crafted to reflect its unique characteristics, as shown in the analysis of waveforms and spectrograms. This systematic approach highlights the versatility and creative potential of audio synthesis in producing dynamic and realistic compositions. Overall, the project showcases a robust understanding of sound design and its practical applications in digital audio synthesis.