Project documentation

B2M31SYN - Synthesis of audio signals

MARTIN NOVOTNÝ

novotm80@fel.cvut.cz

FACULTY
OF ELECTRICAL
ENGINEERING
CTU IN PRAGUE

Introduction

The semestral project for the subject B2M31SYN consists of two main parts

1. obligatory composition synthesis (with instrument scale),
2. own synthesis.

Each part is described in more detail in selected section.

My work is structured in two main directories

- `data` - input data for synthesis (for example lookup table for wavetable synthesis),
- `fcn` - functions used for synthesis (each source has its own function).

The whole synthesis can be started by running script `main.m`, the output audio files are then saved into the folder `results` (folder `private` contain functions for MIDI toolbox).

I tried to use the function skeletons from the lessons and perform slight improvements to get „the best“ sound while keeping the simple basics. My goal also was to try each synthesis type (or at least majority of them) we have talked about - that is why I have chosen the synthesis of unmusical sources in own synthesis. To simplify my work, I decided to unite the input parameters of sound source functions - the parameters are as follows

- `freq` - frequency of synthesised signal (only used for musical sources),
- `dur` - duration of synthesised signal,
- `fs` - sampling frequency od synthesised signal.

I hope that each funtion code is sufficiently commented so I do not have to analyze it more. If you have any questions, please email me.

Obligatory composition synthesis

For the obligatory composition I have chosen the famous ABBA song Waterloo. The supplied MIDI file for this song consist of following tracks (together with the synthesis type I finally used for the synthesis)

- **vocal A** - formant synthesis (with vibrato and tremolo),
- **saxophone** - formant synthesis (with vibrato and tremolo),
- **trumpet** - FM synthesis,
- **grand piano** - KS algorithm,
- **electric guitar** - KS algorithm,
- **bass guitar** - KS algorithm.

When creating those instruments, I tried multiple methods of synthesis (taking into account the results from individual lectures) and in the end, I have chosen the one from which the output audio sounded the best. I did not try to exactly match the spectrums of examples, but I focused more on the audio output. In many cases, the most demanding part of the synthesis was the envelope of signal. Fortunatelly, I downloaded some samples from [1] and using an envelope detector I tried to figure out the envelope shape. I have mostly used ADSR and exponential envelope.

For the scale output, I have used traditional C major scale from C_2 to C_5. Length of each tone is 0.9 s. The order of instruments is following

1. **bass guitar** - notes C_2 to F_2,
2. **grand piano** - notes G_2 to C_3,
3. **vocal A** - notes D_3 to G_3,

4. **electric guitar** - notes A_3 to D_4,
5. **saxophone** - notes E_4 to G_4,
6. **trumpet** - notes A_4 to C_5.

At the end of synthesis, I added small reverb using IIR filter (not to scale) and made pseudostereo file using Hass effect. The time delay between two output channels is 13 ms.

Own synthesis

In my own synthesis I tried to compose easy foley story. The abstract of the story is simple - during bad weather (which is symbolised by thunders, wind and rain) a man sits in his car, starts engine and drives away. During driving he decides to call a close friend (DTMF sound). While he is trying to contact the friend, he does not pay much attention to driving, which leads to a traffic accident.

I have selected this topic, because I wanted to try synthesis of unmusical sources, which could be connected together. The own synthesis consist of following sources (respectively MATLAB functions)

- `bells` - church bells; additive synthesis,
- `boom` - car crash sound; subtractive synthesis,
- `destination` - final destination sppech; vocoder,
- `doorClap` - car door closing; subtractive synthesis,
- `DTMF` - multi-frequency selection; additive synthesis,
- `engine` - car engine sound; wavetable synthesis,
- `scream` - driver scream before crash; formant synthesis (with vibrato and tremolo),
- `thunder` - thunder crash; subtractive synthesis,
- `warningTone` - before-crash warning sound; additive synthesis,
- `wind` - wind and rain sound; synthesis mixture.

Results

To be honest, I am really pleased with the results, especially with the guitars. I found out, that the nonlinearities really play part and helps the output sound more like guitar. In my opinion the worst result is for the trumpet, especially at the high pitch when playing together with saxophone (for example at 01:25 of obligatory composition). I am not quite sure, where the problem is, but I hope that after more research (or by using more sophisticated codes) the sound would be better.

When I was performing the synthesis of unmusical sources, I found out, that the synthesis was more creative, because I had more degrees of freedom and I could „play“ more with each synthesis parameters. On the other hand, when I tried to get the output audio sound like non artificial sound (for example such as wind) it was often very difficult. The difficulties, in my opinion, came from the complexity of spectrum and envelopes of those signals. The output of own synthesis is also very influenced by demanding mixing and mastering process in MATLAB - maybe in ProTools or similar software I could make the final mix sound better (for example with adding panning of selected sources and the like).

Literature

[1] R. Čmejla: B2M31SYN – Syntéza audio signálů [online]. Available: http://sami.fel.cvut.cz/syn/

[2] Digital audio effects [online]. Available: http://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf

[3] FM Synthesis for Musical Instruments [online]. Available: http://home.eng.iastate.edu/julied/classes/ee224/Labs/FMSynthesis_lab.pdf

[4] Formant analysis [online]. Available: https://ccrma.stanford.edu/ jmccarty/formant.htm

[5] Digital audio synthesis [online]. Available: http://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/PDF/05_Audio_Synthesis.pdf