Recognize Note Block sound samples in a music file

Independent laboratory

Extended Abstract

Last modification: 12/06/2024

Contents

General information

Topic description

The goal of the project is to create a command-line conversion program that can recognize certain sound samples from music in the form of a wave audio file (wav/mp3) and create an NBS file based on it (the own format of the Note Block Studio program). Basically, the goal is to recognize patterns in sound files containing Note Block patterns, at different times and pitches, with good results. The problem is very similar to the task of recognizing a MIDI file from a wave file, it can also be considered a special case of it.

Conditions of creation

I am preparing this program as part of the Independent Laboratory subject of the BME engineering informatics course.

I write the program in Rust (I use version 1.75.0), for file formats and Fourier transformation I use some of the crates in the community library, which the package manager can automatically download before compilation.

The source code is available under the MIT license on Github.

Introduction

About the wave to midi problem in general

Recognizing sounds in wavy sound files is a common task, and there are already many solutions and programs for it. Such a program can be used, for example, if a composer wants to compose his music on a real instrument, but at the same time wants to post-work with digital tools. Or someone wants to write down an already finished piece of music, either so that it can be played, or so that it can be visualized. On YouTube, for example, these so-called piano tutorial videos, which people mostly watch not to learn to play the piano, but because they look good. One of the pioneers in this field was a program called Synthesia, so this type of video is often called synthesia.

A special case of this, wave to nbs

This project deals with a more specific, special case, where we are looking for pre-known sound samples in the sound file, whose pitch and length can be known quite precisely, and the samples are also known in advance. For this reason, in theory, much more precise recognition is possible than before. So far, the best automated approximation has been if we converted the wave audio file to MIDI using one of the MP3 (or other wave format) to MIDI converters available on the market, and then using a program called Note Block Studio (the format of which is our target format, nbs) we imported the MIDI file. None of the conversions are lossless, but especially the wave to MIDI part is imprecise, that is, it could be much more accurate, knowing the samples.

Motivation

This would be beneficial in several ways, for example, with the knowledge of the sounds, you can make a visualization of the music such, which would not be possible without the sounds, only with the knowledge of the wavy sound file. Another possible use is if someone wants to rework the music, re-arrange it. For this, you also need to know what sounds are heard at which moments in time. Here there is, for example, a playlist of arrangements based on Note Block music.

Outline a solution

During recognition, we first determine the moments in time that are worth examining more closely, i.e. at least one pattern will probably start to sound at that moment in time. This can be, for example, a sudden increase in volume because each sample is loud at the beginning and then fades away. We then compare all the pitches of all the known samples with the fraction of a second of the music to be recognized, and determine the instrument-pitch combinations that are worth further investigation and those that definitely do not sound. Then we run a library Nelder-Mead optimization with the potential sounds, which minimizes the sum of the absolute value of the difference between the potential sounds and the target. Then we get a volume for each of them, which is around 0, if the minimum search came up with no sound there. Then these sounds will be our tips for that moment in time.

Related works

Open Note Block Studio

GitHub link. Editor of the target format (nbs), you can use it to edit, create, view, export Minecraft Note Block music as a wave sound, and also as a redstone circuit that can be played in Minecraft (although this part needs revision from several aspects).

Nbswave

GitHub link. ONBS's new wave exporter, because the old one is quite imprecise and the overclocking is not solved either.

Search method

Pre-recognition

In order to minimize the state space of the library optimization algorithm, we try to recognize in advance those instrument-pitch combinations where sound is conceivable and those where it is not. This is done by subtracting the spectrogram of the sound sample from the spectrogram of the piece of music to be recognized (per element), and then using a ReLU-like (f(x)=max(0,x)) function, per pixel (i.e. for each given time-moment-frequency pair). Then the "errors" (or their square) are added up, and based on this we decide with a threshold whether to accept the possibility of the existence of the given sound. This ReLU is needed so that we don't penalize if there are many sounds in the piece of music, but penalize if the exact frequency that should be there isn't there.

Optimization

After determining the pair of instrument-pitch pairs that potentially sound in the given tick, we use a library optimization algorithm to determine the state corresponding to the minimum of the error, where the error is the sum of the absolute values of the difference between the two spectrograms (~MAE), and the state is the volume per note , mostly between 0 and 1.

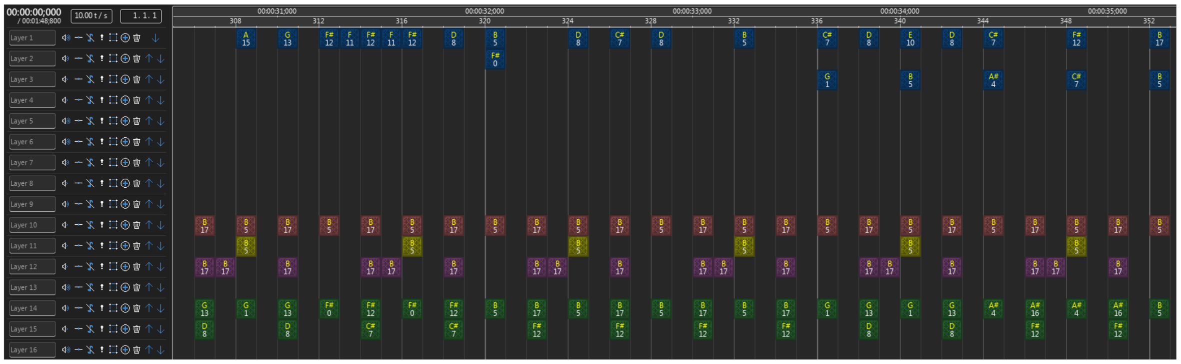
Among optimization algorithms, I tried Particle Swarm Optimization and the Nelder-Mead method, of which the latter seemed better, I will explain this below. They were mentioned because the error is the result of a complex calculation, which cannot be derived, or at least is very complicated. The essence of the Nelder-Mead method: in the case of n dimensions, it defines an n-dimensional tetrahedron with n+1 ”vertices”, then transforms the vertices step by step: one away (increased, reduced, or projected), or brings the vertices closer to each other, etc., depending on what error you get from the function we provide at those points.

Results

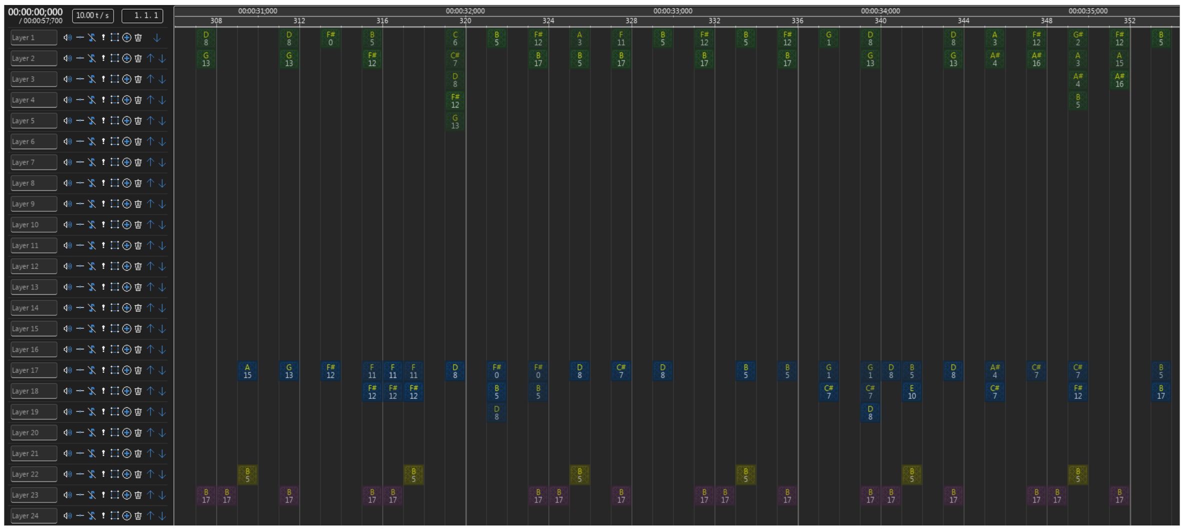
General rating

The program can quite successfully recognize music exported with nbswave, which may contain harp, bass, snare and click instruments. On the other hand, it is necessary to specify the degree of compensation against overdrive and the tps in order to have a good recognition, it would be good to recognize these later by default. Further improvement of the recognition and testing of the program with more music, and later with several instruments, is necessary. Testing on sound files obtained with other exporters and recording methods is also necessary: better protect against over-control, small pitch shifts, and inaccuracies in timing. The program is able to take advantage of multithreading and runs in a reasonable amount of time using compile-time optimization, although additional runtime optimization would make sense. On my 8-9 year old intel laptop, it currently recognizes 1 minute of music in about 2 minutes.

Original:

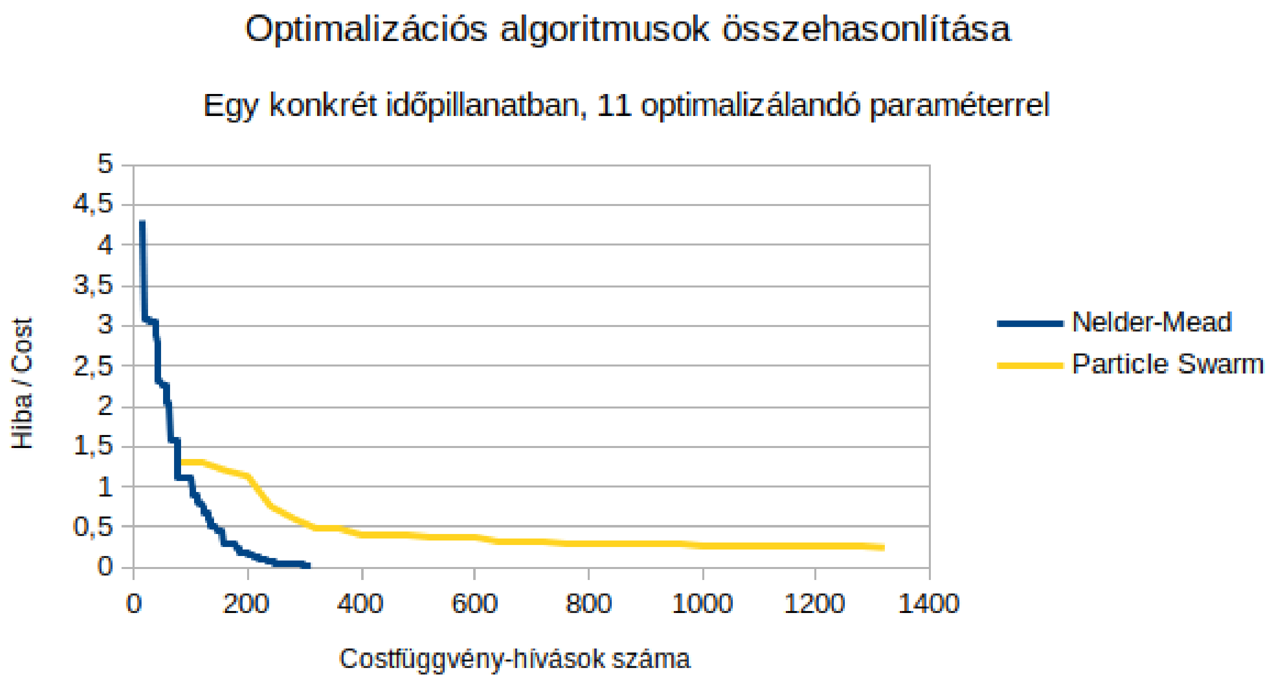


Recognized by:



Comparison of optimization algorithms

The figure below compares the two tested methods (Nelder-Mead Method and Particle Swarm Optimization) based on the number of cost function calls and the error achieved.



Of course, the use of both methods could be further optimized, with better initial values, a better swarm size than PSO, and fine-tuning of the parameters. But basically, Nelder-Mead was much faster and gave a better result. Moreover, even if I continue to run PSO with these parameters, it cannot give a better result than the error shown in the graph.

I ran 200 iterations for both methods, although this means different things for the two methods. With its 311 function calls, NM finished in 1.607s (=193.5 calls/s) and achieved a minimum cost of 0.02367, while PSO with its 8040 function calls in 39.295s (=204.6 calls/s) 0 , reached a minimum cost of 2434. So, in addition, between the calls, Nelder-Mead thought more, which is beneficial for this problem, since it is quite expensive to call the cost function.

The values found were as follows:

Solution	No	There is	No	No	There is	No	No	There is	There is	There is	There is
NM	0.007655	0.7186	-0.0072	-0.00057	0.7188	0.001538	-0.0012	0.7197	0.7247	0.7165	0.7175
PSO	0.0	0.7189	0.0	0.0	0.7188	0.0	0.0	0.7184	1.0	0.7165	0.7195

Summary

Overall, we successfully analyzed the wavy file, and found strong hints for solo voices at certain moments. We got to know the nbs format and the special features of Note Block music.

In the future, the implementation of various developments would be beneficial.