AMPLAB - Sound retrieval with Freesound and creative applications
T. LE ROUX - 30/03/2023

Goal :

In this assignment, we are going to do audio-mosaicing. Audio mosaicing is a technique that involves analyzing an audio signal and dividing it into small segments. Each segment is then replaced by a similar segment from another audio recording, creating a "mosaic" of different audio sources.

We are therefore going to have 2 different types of audio files :

- Target file : The file that we are going to try to reproduce with portions of the “resources files”.
- Resources files : A group of audio files that we have to analyze and cut in order to recreate the target file.

We have 3 notebooks for this assignment :

1. The first one is dedicated to creating the sound collection for the resources files
2. The second one is dedicated to selecting a target file, and analyzing all our files using essentia
3. The third one is dedicated to the reconstruction of the target file.

Our project is going to try to recreate an iconic track by british electronic music producer Leon Vynehall : Midnight on Rainbow Road (https://www.youtube.com/watch?v=M4zSy07qPkI&ab_channel=RushHour) , that was made for Gerd Janson’s Musik for Autobahns 2, a compilation of ambient pieces of music aimed at mind traveling while driving (‘Ambient Race Car Music’).

The initial BPM calculated by RekordBox is 109, and we slow it to 80BPM using Audacity.

This track is very ambient, with pads, a simple piano melody and some rare light percussions.

First Notebook :

In this notebook, we gather the resources files using the Freesound API.

We do the following queries :

- ‘piano drone’ from 0 to 30s
- ‘ambient atmosphere’ from 0 to 30s
- ‘dreamy pads’ from 1 to 30s

…

Second Notebook :

In this notebook, we extract the feature from all our sounds.
For this, we have to select a frame size and also a bunch of features.

For the frame size :
We’ve gone from 109 BPM to 80BPM.

This means that there is 60/80 = 0.75s per beat, therefore 44100*0,75s ~= 32768 samples = 1 beat. Therefore, 8192 = 1/4th of a beat.

Even if the track is ambient, there are still some rhythmic elements, therefore we should not lose this information in the final results.

Regarding the features, we change the base code to extract way more features. We extract a subset of 110 descriptors from low level, rhythm and tonal. We restrict our values to only float values.

```
if 'metadata' not in descriptor and isinstance(features[descriptor], float)
```

```
features, features_frames = estd.MusicExtractor(lowlevelStats=['mean', 'stdev'], #MusicExtractor
                                                rhythmStats=['mean', 'stdev'],
                                                tonalStats=['mean', 'stdev'])('tmp.wav')
```

To make this change, we have to save every frame of 8192 samples to a temporary file (called tmp.wav). In order to make sure that we have something to extract and that the frame is not silent (which would trigger an Essentia error), we have to assign the first value of the frame to 0.1, which might slightly change our data (but should not have a too big impact).

The extraction takes nearly 30 minutes for our 90 resource files and our target file.

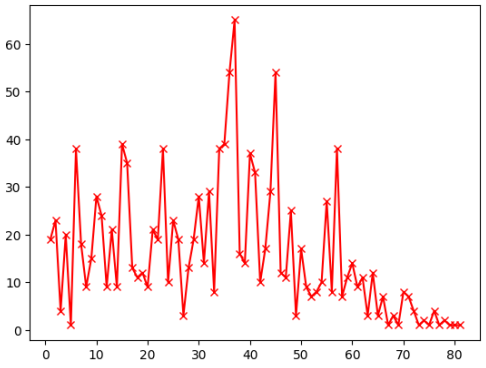
Third Notebook :

In this notebook, we realize the reconstruction.

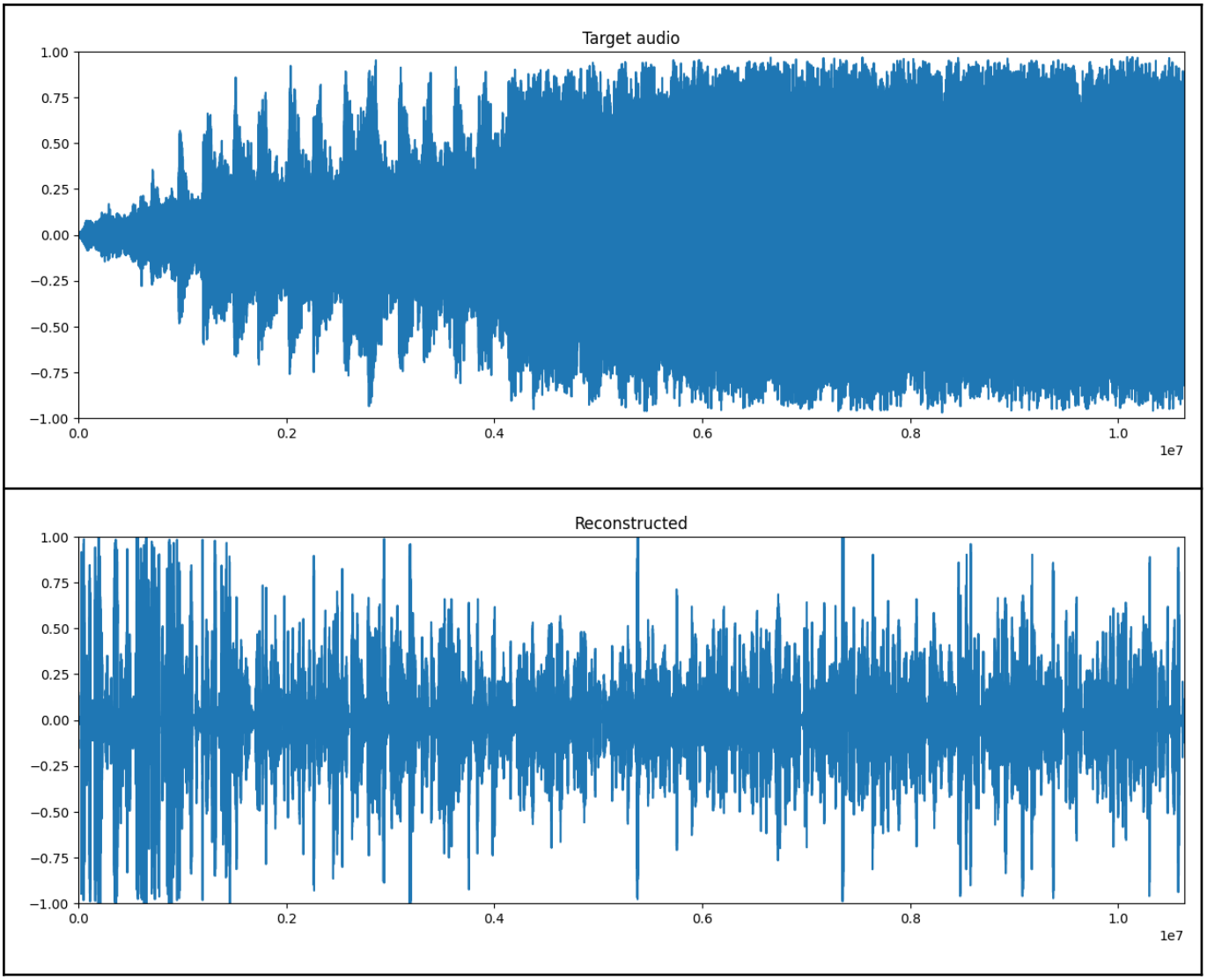
Because we added some features in the extraction, we want to consider them for the reconstruction. Therefore, we concatenate the ‘basic’ features given in the original notebook with our new features.

The computing for the reconstruction takes a longer time (from 30s to nearly 3mn), but we should have a result way closer to what we had before.

We can plot the number of times that the various resources files have been used in our reconstruction, and we can see that we used a lot of them (81 out of 89) and that some of them are really frequently played.



We find very different waveforms :



When we listen to the reconstructed track, it's really really disappointing. The ambient side of the track is lost. Since the track is not rhythm based, but follows a melodic soft progression, this side is completely lost, and we find this mess of elements not sounding well together, and creating a rhythm where there should not be one.

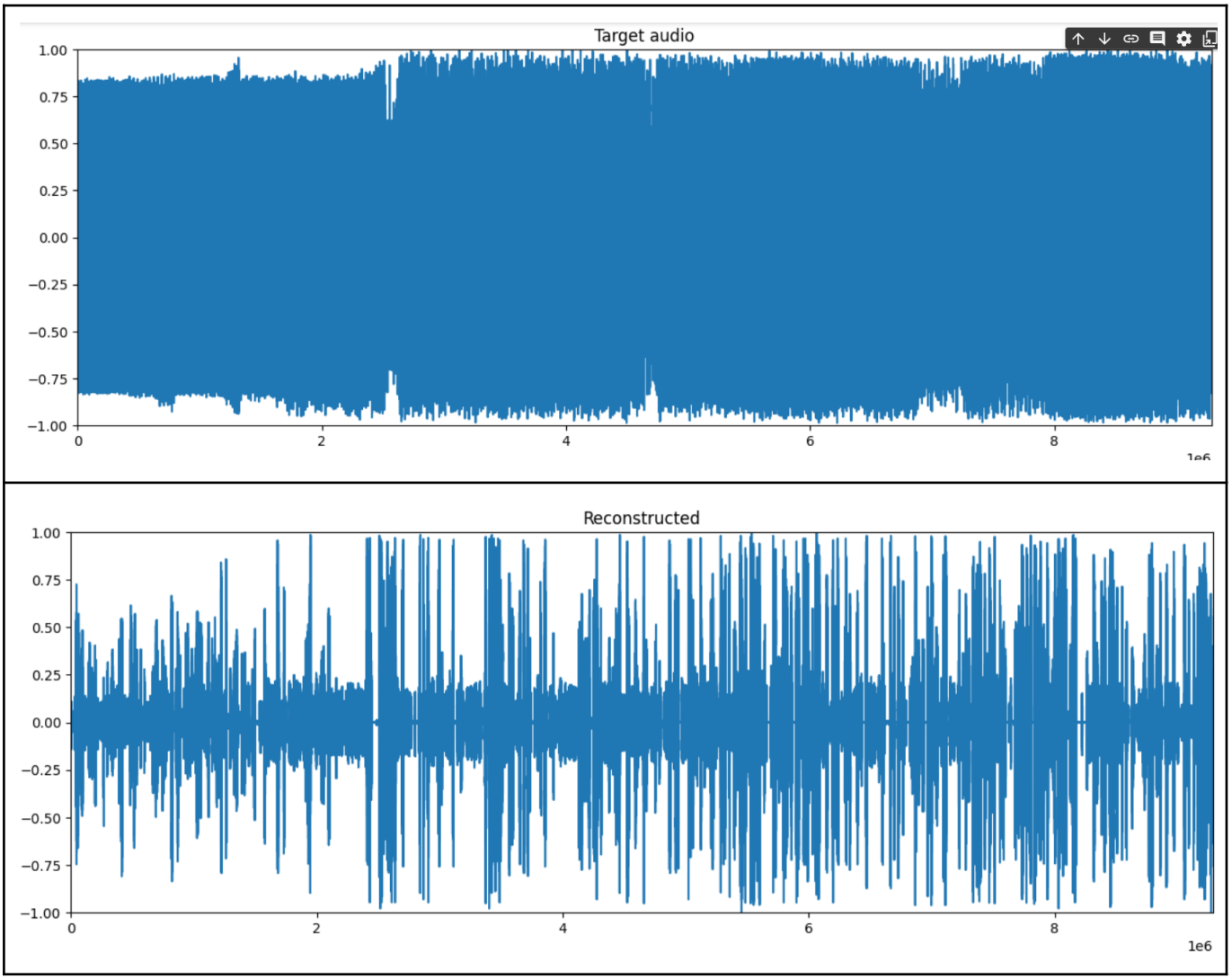
It should, however, work way better for a highly rhythmic techno track. The repetition of carefully picked elements should make it more interesting and coherent.

Testing with another track :

We therefore try with a second track, way more rhythmic, really fast techno from Sidewinder (aka D.Dan) (https://www.youtube.com/watch?v=Bc9qUWt6jxs&ab_channel=OOUKFunkyOO). It contains really bright percussions, and is melodic. We move the BPM from 155 to 160, based on our formula, we should have 1 frame every ½ of beat.

We don't change the first analysis of the resources files, and we do another dataframe_target.csv for this one (simply named dataframe_target2.csv)

We reconstruct and have the following waveforms :



They look once again very different.
When we listen to the reconstructed audio, it's also not incredible. Because we don't have a clear kick in our sounds (only piano / pads / …), the reconstruction automatically tries to find a kick. It finds a weird kick, and it doesn't sound really well. Also, it jumps very often. Maybe the choice of a window size of 8192 was not the best, however dividing this window by 2 would multiply by 2 the calculation of our features.
The result is still better than the ambient track, but it could still largely be improved.