Giray Eryılmaz

Q1:
Code is straightforward first I draw spectrograms to see what I am dealing with.
Than I take Fourier t. to switch to frequency domain, again I plot
Than I plot in the time domain
After deciding on some frequency band I filter, (I actually tried a few different combinations on low and high stop freq.s)
Than I plot frequency and time domain representations of original mike and recovered mike to see how good the filter did . Also I calculate the snr too for the same reason.

1_1-it appears street has a larger variety of frequencies, has high frequency components but there is region mike and street collides.

1_2- mike is more “discrete” street is a wider sound, and in the mixed one street “fills the gaps”

1_3- they are similar but still recovered one seems to have o wider range of freq.
They are not obviously identical.

1_4- here we clearly, without a doubt lost some data but with such a filtering system it was inevitable.
Also we failed to filter out some noise in rest of the freq range which also is natural

rns is 2.5226 which is positive, we can somewhat increase this but when I do and listen the recovered sound, although rns gets better sound gets worse (in my opinion at least).

Q2:
First I would like to tell what I did,
I draw spectrograms of both given clap and slap sounds.
I searched for differences between them, and noticed that unlike clap , snap has more high frequency components so I decided to differentiate these by that property.
17.000 seemed a convenient threshold.
To see if given sound has components in frequencies higher than the threshold,
I take fast fourier transform to get frequency representation, I use

```
th_fr = 17000;

baseFreq = (2/ fs) * N_sound;

n_1 = int64(th_fr*baseFreq);
```

to get where the index of given frequency is.

Then calculate power using abs, and max to find largest power component above threshold frequency. If it is lager than 0.0006 I accept it to be a clap.
I now that this is a primitive solution but a professional solution would require machine learning with lot of data.

Q3:

Code is quite straightforward, I apply the given filtering and store calculated snr values on lists. At the end I plot those arrays to see how they go.

3_1- taking a = 0 makes no change and taking it larger clearly destroys the original sound.

3_2-by the nature of the n_tap filter first “iteration” (I dont know what to call it) makes a big change in the signal, second one reduces it than third one, the signal ‘oscillates’ and after a point it settles
which makes sense.

3_3- bigger delays seem to be less effective, they do not cause big disruptions which is again reasonable imagine making the delay larger than the length of the signal, we would not even feel it when we look at the original length of the resulting signal.