Mobile Denoiser
with Pure Data and MobMuPlat

Pol Valls
NIA 184218
Audiovisual Systems Engineering
Universitat Pompeu Fabra

David Mitjana
NIA 182486
Audiovisual Systems Engineering
Universitat Pompeu Fabra

***Abstract*—Nowadays noise is very present in our daily life, from the hum of the air conditioning, washing machines or any electronic device, street noise, crowd noise, etc. As a consequence , there is an increasing interest and adoption of noise reduction techniques, ranging from noise cancelling headphones to isolation windows and noise cancelling systems for cars. With this in mind, our aim has been the creation of a mobile denoiser app capable of attenuating ambient noise by taking advantage of the powerful audio processing tools that the Pure Data environment [1] brings. As a result, for our work we have used MobMuPlat[2], a software suite for running Pure Data patches on mobile devices. Long story short, we have built two variations of the same app: one to apply noise reduction to our recordings (i.e. quick recordings, voice notes), and another one to apply real time noise reduction for live listening.**

***Index Terms*—Real time audio processing, Pure data, MobMuPlat, Noise reduction.**

I. Introduction
Problem statement and background

First of, as a conceptual background, noise reduction technologies and methodologies were very attractive and appealing to us. Consequently, our initial idea was to build an Active Noise Cancelling (ANC) system with pure data. However, we soon realized this implied a number of drawbacks we could not easily deal with, since the creation of a destructive interference between an external noise and an adaptive waveform exactly at the listener's ear involved dealing with waveform and power detection followed by a very delicate phase shift which would require an integrated hardware and software design[3].

Because of this, after giving it a little thought, we ended up adapting our initial idea to a more reasonable goal. That is why we followed the idea of building a mobile oriented denoiser using the I04.noisegate.pd patch from the Pd documentation as a starting point. This patch is an approach to a noise gate, and allows noise suppression with a **narrow-band companding approach**. With this patch, the user can analyse a noise, generate a frequency based mask of this noise and then apply the mask to a combination of the desired sound track plus some noise, effectively subtracting the noise component. Moreover, it also allows for a tuning in terms of the amount of masking applied (mask level). Looking into the technical background, this patch (and therefore our work as well) is mainly based on an analysis and synthesis of the signal based on the Discrete Fast Fourier Transform, meaning that the mask is calculated in the frequency domain with an estimate of its average power and then subtracted from the sound before being synthesised back to the temporal domain again. [4][pag. 290].

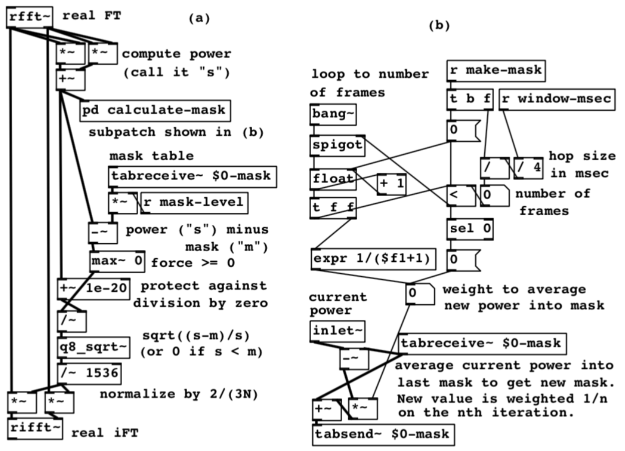

Fig. 1. The Pure Data's *noisegate.pd* fftanalysis subpatch object scheme. Image from [4]. (a) analysis and reconstruction of the signal; (b) computation of the "mask".

In this paper we will describe the process we have followed to adapt the *I04.noisegate.pd* patch to create our two denoiser apps with the MobMuPlat software.

II. Description of the system

The pd patch *i04.noisegate.pd* provided a very solid basis to start building our mobile denoiser. Nevertheless, we needed to implement several changes in order to adapt it to a portable solution with our desired features.

The key point features of usability that we have implemented for our app are the following:

- Capacity to record sound.
- Include a play/stop capability and interface.
- Generate the mask when desired.

- Control over output volume.
- Real Time Adjusting of mask-level.
- Real Time ON/OFF switch of the noise attenuation process.

We will see later that the GUI of the app has custom objects to control all of the above features.

For the recording feature we simply use the input object [adc∼] connected to a [tabwrite∼] object, controlled by a *startREC* and *stopREC* bang messages that come from the GUI. In addition, this bang messages are useful to estimate the duration of the recording. The maximum duration of the audio file is set to 10min, but the size of the table (also known as array) containing the recording will adapt to its computed duration.

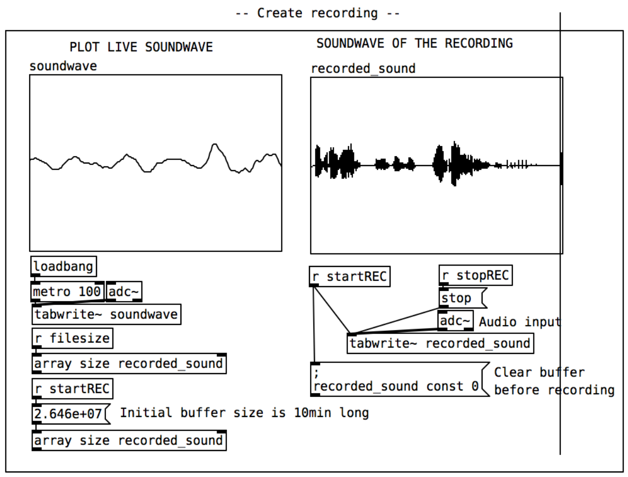

Fig. 2. This is our Pd implementation of the sound recorder.

In addition, we have made it possible to calculate the mask when desired with a copy of the *fft-analysis* subpatch and a make-mask message that is sent when the user indicates. We have also set the mask level within a range between 0 and 25.

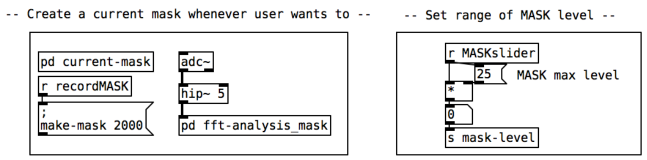

Fig. 3. This is our Pd implementation of the users make-mask option.

Finally,there is a [tabplay∼] object connected to a users start/stop message, which is redirected directly to the output or through the noise reduction engine (a version of the fft-analysis subpatch which does not udate the mask). Furthermore, our apps are only suitable for spoken voice, not music, and this implies that we will not need to work with high frequencies as we would have in a music oriented app, so generally we will have smaller bandwidths. Thus, it is not necessary to use the standard but high sampling rate of 44.100Hz (which following the Nyquist-Shannon theorem is defined as twice the maximum frequency that can be represented). Thefore, we consider that a sampling rate reduced to the half (22.050Hz) is enough to record, analyse and synthesise voice.

III. Results

The final look of our apps is the following [5]:

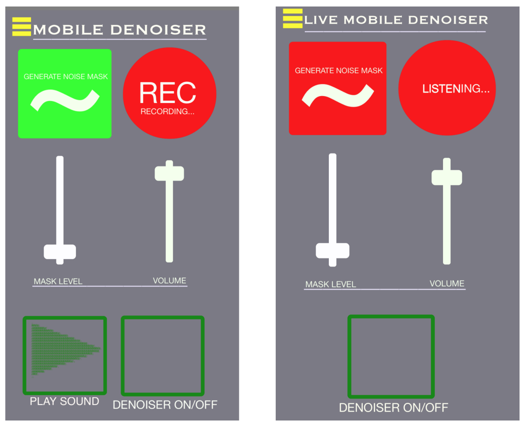

Fig. 4. This is our Pd implementation of the users make-mask option.

IV. Potential future work

Since a mobile denoiser it is a very useful and powerful tool, many other adaptations and improvements can be made from the point of view of UX/UI, usability and features. Some of the ideas we would like to have implemented or tested are the following:

1) Compute a mask automatically and adaptively during the actual recording of the sound. . The basic improvement would allow to create the mask as soon as the user opens the app, then detect pauses in the recording and use them to recalculate the noise mask and apply it again. This could be specially useful for long recordings where the noise might change.
2) Once the recording is made, give the option to download or ’save on the device’ the denoised sound.
3) To be able to change form the app advanced option the size of the fft window to values other than 1024.
4) Make a web-based denoiser where the user could upload a recorded sound with noise and download a denoised version of it.

References

[1] M. Puckette, “Pure data: another integrated computer music environment,” in *in Proceedings, International Computer Music Conference*, 1996, pp. 37–41.
[2] D. H. de La Iglesia, “The mobility is the message : the development and uses of mobmuplat,” 2016.
[3] S. M. Kuo, S. Mitra, and W.-S. Gan, “Active noise control system for headphone applications,” *IEEE Transactions on Control Systems Technology*, vol. 14, no. 2, pp. 331–335, March 2006.
[4] M. Puckette, *The Theory and Technique of Electronic Music*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2007.
[5] P. D.Mitjana, “pd-denoiser,” https://github.com/mitji/pd-denoiser.