Voice Processing

Marchioro Thomas

Project 5 – Speech Enhancement

The goal of this project is to find a good approximation of a track $x[n]$ from a noisy version $y[n]$, which is created by adding synthetic white Gaussian noise, i.e.,

$$y[n] = x[n] + b[n], b[n] \sim \mathcal{N}\left(0, 10^{-\frac{\text{SNR}}{20}}\frac{E_x}{E_b}\right).$$

1 Unsupervised Speech Enhancement

Spectral subtraction I implemented the spectral subtraction by estimating the noise PSD $S_b(\omega)$ from the first 1000 samples (known for containing silence). Then, I processed the entire track frame by frame, computing the new short-time PSD as $S_{\hat{x}}(kL,\omega) = \max(S_y(kL,\omega) - S_b(\omega), 0)$, and the corresponding short-time Fourier transform as

$$\hat{X}(kL,\omega) = \sqrt{S_{\hat{x}}(kL,\omega)}e^{j\angle Y(kL,\omega)}.$$

Actually, I found that better results where obtained by subtracting a scaled version of the noise PSD, i.e., $S_{\hat{x}}(kL,\omega) = \max(S_y(kL,\omega) - \lambda S_b(\omega), 0)$, where $\lambda = 30$.

Unsupervised Wiener filtering I implemented an "unsupervised" version of Wiener filtering technique, again by processing the audio track frame by frame, computing for each frame an approximation of the Wiener filter as

$$\hat{H}_s(kL,\omega) = \frac{S_{\hat{x}}(kL,\omega)}{S_{\hat{x}}(kL,\omega) + S_b(\omega)}$$

where $S_{\hat{x}}(kL,\omega)$ is computed using spectral subtraction with $\lambda = 50$.

Estimating the PSD of the noise The PSD of the noise can be estimated as $|B(\omega)|^2$. Nevertheless, if the distribution is known to be white Gaussian, the PSD is constant

$$S_b(\omega) = \sigma^2, \forall\omega$$

and hence a better approximation of the PSD is given by $\hat{\sigma}^2$, which is obtained by estimating the variance of the noise.

2 Supervised Speech Enhancement

Naive inverse filtering A trivial idea to remove the noise having access to both the noisy track $y[n]$ and the original track $x[n]$, is observing that

$$X(\omega) = Y(\omega) - B(\omega) \Rightarrow \frac{Y(\omega)}{X(\omega)} = 1 - \frac{B(\omega)}{X(\omega)} = H_{\text{naive}}(\omega) \Rightarrow X(\omega) = H_{\text{naive}}(\omega)Y(\omega).$$

Obviously, this filter applied to the noisy track $y[n]$ provides an almost perfect reconstruction of $x[n]$, where the only "noise" component is given by the loss of resolution due to the FFT and IFFT processes. Nonetheless, when a new noisy version of the track $y'[n]$ is filtered with $H_{\text{naive}}(\omega)$ the results are poor. The reason is that, differently from the PSD, the FFT itself does not provide a statistical description of the noise.

Method	Music SNR (dB)	Voice SNR (dB)
Spectral subtraction (PSD)	14.84	15.64
Spectral subtraction ($\hat{\sigma}$)	19.51	21.19
Unsupervised Wiener (PSD)	14.97	14.92
Unsupervised Wiener ($\hat{\sigma}$)	19.67	21.24
Naive inverse filter	11.52	11.61
Wiener	20.96	22.70

Table 1: SNR of the reconstruction $\hat{x}[n]$ using different techniques, divided in unsupervised and supervised.

Wiener filtering The Wiener filter is simply computed as

$$H_s(\omega) = \frac{S_x(\omega)}{S_x(\omega) + S_b(\omega)}$$

where in this case $S_b(\omega)$ is computed as the PSD of $y[n] - x[n]$, while $S_x(\omega)$ is the PSD of the original signal $x[n]$.

3 Results

I executed all the methods on both the given music track, `furelise-1000z.wav`, and on a voice track of mine, `marchiorot-fish.wav`. I preprocessed my voice track by repeating it enough times to obtain $\simeq$ 300000 samples and zero padded the first 1000 samples, that are used to estimate the noise. I applied the supervised techniques on a different noisy track than the one used to compute the filters. As expected, Wiener filter (in the original supervised version) gives the best performance, but also the results given from spectral subtraction are reasonable.