Ephraim-Malah Algorithms

Notes

MMSE short-time spectral amplitude estimator based on modelling speech and noise spectral components as statistically independent Gaussian random variables

Time index p
Frequency index w_k
Spectral gain $G(p, \omega_k)$ applied to each short-term spectrum value $X(p, \omega_k)$ given by:
$G = \frac{\sqrt{\pi}}{2} \sqrt{\left(\frac{1}{1+R_{post}}\right)\left(\frac{R_{prio}}{1+R_{prio}}\right)} \cdot M\left[(1+R_{post})\left(\frac{R_{prio}}{1+R_{prio}}\right)\right]$

M stands for the function

$$M[\theta] = e^{-\frac{\theta}{2}}\left[(1+\theta)I_0\left(\frac{\theta}{2}\right) + \theta I_1\left(\frac{\theta}{2}\right)\right]$$

The gain depends on the two parameters R_{post} and R_{prio} which have to be calculated in each short-time frame p and for each spectral component ω_k

R_{post} = the a posteriori signal-to-noise ratio (SNR) = estimate of the signal-to-noise ratio in the current short-time frame
$R_{post}(p, \omega_k) = \frac{|X(p,w_k)|^2}{v(\omega_k)} - 1$
$v(\omega_k)$ = noise power at frequency ω_k

The a priori signal-to-noise ratio R_{prio} is defined as
$R_{prio}(p, \omega_k) = (1-\alpha)\mathrm{P}[R_{post}(p, \omega_k)] + \alpha\frac{|G(p-1,\omega_k)X(p-1,\omega_k)|^2}{v(\omega_k)}$
$\mathrm{P}[x] = x$ if $x \geq 0$, $\mathrm{P}[x] = 0$ otherwise

$|G(p-1, \omega_k)X(p-1, \omega_k)|^2$ denotes the spectral of the noise-reduced output in the last time frame - corresponds to an estimate of the SNR in the time frame with index $p-1$

R_{prio} is therefore an estimate of the SNR that takes into account the current short-time frame with weight $(1-\alpha)$ and the noise reduced previous frame with weight α

- α set to 0.98 in previous papers (Ephraim and Malah, 1984; Cappe, 1994)

The a priori SNR is the dominant parameter

- Strong attenuations only obtained if R_{prio} is low
- Low attenuations only obtained if R_{prio} is high

The a posteriori SNR acts as a correction figure whose limited to the case where the a priori SNR is low

- The larger the R_{post}, the stronger the attenuation

The a priori SNR is evaluated by the nonlinear recursive function shown above

- When R_{post} stays below or is sufficiently close to 0 dB, the a priori SNR corresponds to a highly smoothed version of the a posteriori SNR over successive short-time frames
  - The variance of R_{prio} is must smaller than the variance of R_{post}
- When R_{post} is much larger than 0 dB, the a priori SNR follows the a posteriori SNR with a delay of one short-time frame
- When the a priori SNR is high, the attenuation brought to the spectrum is negligible
  $R_{prio}(p, \omega_k) \approx (1-\alpha)R_{post}(p, \omega_k) + \alpha\frac{|X(p-1, w_k)|^2}{v(\omega_k)}$

As $R_{post}(p, \omega_k) \gg 1$ in this case,
$R_{prio}(p, \omega_k) \approx (1-\alpha)R_{post}(p, \omega_k) + \alpha R_{post}(p-1, \omega_k)$
As $\alpha \approx 1$, we can write
$R_{prio}(p, \omega_k) \approx \alpha R_{post}(p-1, \omega_k)$

Choice of α is guided by a trade-off between the degree of smoothing of parameter $R_{prio}(p, \omega_k)$ is noisy areas and the acceptable level of transient distortion brought to the signal

- When the analysed signal contains only noise at a given frequency, both the average value and the standard deviation of the a priori SNR are proportional to $(1-\alpha)$, when α is sufficiently close to one
  - Counter musical noise effect (distortion effect) by choosing values as close to one as possible
- When a signal component appears abruptly, the algorithm reacts immediately by raising the gain from a low value to a value close to 1 iff the SNR of the signal component is larger than $1/(1-\alpha)$
  - For signals with lower SNR, R_{prio} takes longer to reach its final value
    - Unwanted attenuated of low amplitude signal components for higher values of α

Approximate limit of $1/(1-\alpha)$ found by considering case where R_{post} is a deterministic quantity before frame index p_0 and has a fixed value of $\mathcal{R}$ for short-time frames with index $p \geq p_0$

- As the gain is null before p_0, $R_{prio}(p_0, \omega_k) = (1-\alpha)\mathcal{R}$
- At $\alpha = 0.98$, R_{prio} is bound to be below or equal to 15 dB
  - 15 dB when $R_{post} >> 1$
  - In the frequency bands containing only noise, $R_{prio} = -15$ dB on average

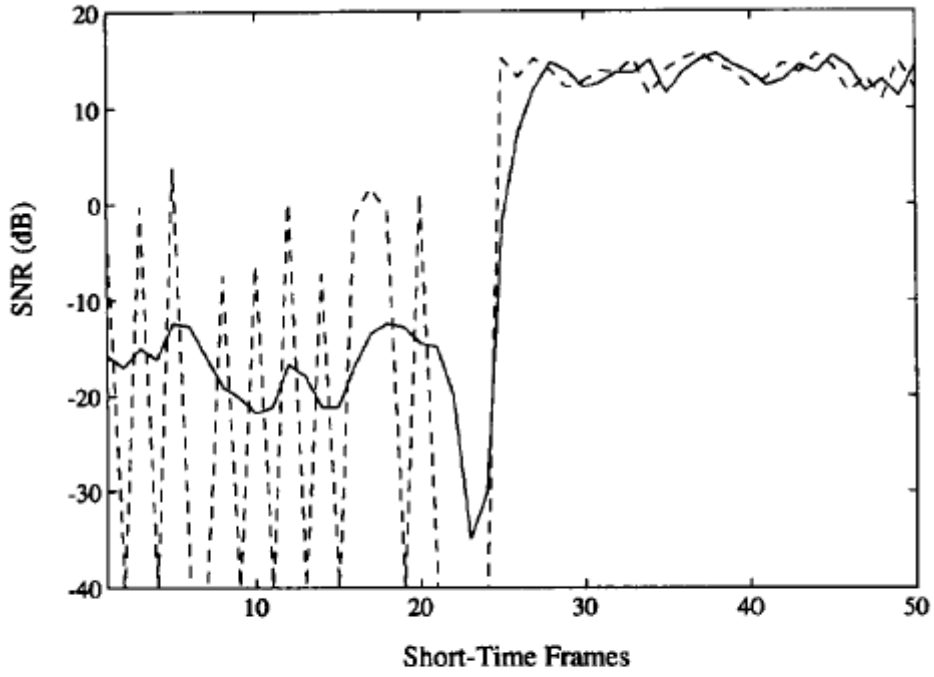

Fig. 3. SNR's in successive short-time frames; dashed curve: *A posteriori* SNR; solid curve: *A priori* SNR. For the first 25 short-time frames, the analyzed signal contains only noise at the displayed frequency; for the next 25 frames, a component with 15-dB SNR emerges at the displayed frequency. Parameter α is set to 0.98.

To prevent musical noise (distortion artifacts), R_{prio} is constrained to be larger than a threshold R_{min}

- R_{min} is chosen to be larger than the average R_{prio} in the frequency bands containing noise only
- In the case where $\alpha = 0.98$, as the average value of R_{prio} in noise only frequency bands is -15 dB, $R_{min} = -15$ dB

Spectograms

In the matrix, time increases across the columns of s and frequency increases down the rows, starting from zero.

- Each column of s contains an estimate of the short-term, time-localized frequency content of data

If x is a signal of length Nx, then s has k columns, where

- `k = [(Nx - overlap)/(window - overlap)]` if window is a scalar.

`s(freq,:)` gives the part of the spectogram for the normalised frequency `freq`

Code

See:
https://uk.mathworks.com/help/signal/ref/spectrogram.html

https://uk.mathworks.com/matlabcentral/fileexchange/10143-mmse-stsa
https://uk.mathworks.com/matlabcentral/fileexchange/27912-vuvuzela-sound-denoising-algorithm

Main

```
% should convert this into function
% need to start out with period without any signal, just noise - e.g. no speech
% default value = 0.25
% Load in audio
[data, fs] = audioread('OSR_us_000_0010_8k.wav');
info = audioinfo('OSR_us_000_0010_8k.wav');
NoSamples = info.TotalSamples;

% Parameters
alpha = 0.98; % standard value in literature
initialSilence = 0.05;
windowLength = floor(0.025*fs); % 0.025 * sampling frequency (in # of points)
% 0.025 gives time taken in s
overlap = .5; % windows overlap by 50 percent (12.5ms)
noverlap = windowLength*overlap; % number of overlapped samples
shiftPercent = 1 - overlap; % how much the window is shifted by each time
window = hamming(windowLength); % creates hamming window

% calculate spectrogram - computes short-time Fourier transform
[s, w, t] = spectrogram(data, window, noverlap);
absS = abs(s);
% s = spectrogram
% w = vector of normalised frequencies
% t = vector of time instants at which the spectrogram is computed

% we want to find the part of the spectogram where it is only noise
initialSilenceSegments = floor((initialSilence*fs-
windowLength)/(shiftPercent*windowLength) + 1);
noisePowerMean = mean(absS(:,1:initialSilenceSegments)')';
noisePowerVariance = mean((absS(:,1:initialSilenceSegments)').^2)';

X = zeros(size(s)); % initialise smoothed signal
gain1 = ones(size(noisePowerMean));
gainNew = gain1;
numberOfFrames = size(s, 2);

NoiseCounter = 0;
NoiseLength = 9; % Smoothing factor for the noise updating

for i=1:numberOfFrames
    % Speech detection and Noise Estimation
    if i<=initialSilenceSegments % If initial silence ignore speech detection
        SpeechFlag = 0;
        NoiseCounter=100;
    else % Else test for speech
        [NoiseFlag, SpeechFlag, NoiseCounter, Dist] = detectSpeech(absS(:,i),noisePowerMe
    end
```

```
    if SpeechFlag == 0 % If no speech found
        noisePowerMean = (NoiseLength*noisePowerMean + absS(:,i))/(NoiseLength + 1); % Up
        noisePowerVariance = (NoiseLength*noisePowerVariance + (absS(:,i).^2))./(1 + Nois
    end

    % Estimate a priori and a posteriori SNR
    Rpost = (absS(:,i).^2)./noisePowerVariance;
    Rprio = alpha*(gain1.^2).*gain1 + (1-alpha).*max(Rpost - 1, 0);

    % Calculate gain
    gainNew = gainFunc(Rprio, Rpost);

    % Replace NaN values for gainNew
    Indx = find(isnan(gainNew) | isinf(gainNew));
    gainNew(Indx) = Rprio(Indx)./(1+Rprio(Indx)); %Wiener Filter

    % Get cleaned value
    X(:,i)=gainNew.*absS(:,i);
end

% Need to reconstruct signal from spectogram
% Use overlap add technique
NFreq = size(s, 1);
ind = mod((1:windowLength)-1,NFreq)+1;
reconstructedSignal = zeros((numberOfFrames-1)*noverlap + windowLength,1);
p = ceil(log2(windowLength));
NFFT = max(256,2^p);

for indice = 1:numberOfFrames
    leftIndex=((indice-1)*noverlap) ;
    index = leftIndex + [1:windowLength];
    temp_ifft=real(ifft(X(:,indice),NFFT));
    reconstructedSignal(index) = reconstructedSignal(index)+temp_ifft(ind).*window;
end

figure(1)
T = 1/fs; %sampling period
t = (0:NoSamples-1)*T; %time vector
tnew = (0:length(reconstructedSignal) - 1)*T;
title('Original Signal')
subplot(2,1,1);
plot(t, data)
subplot(2,1,2);
plot(tnew, reconstructedSignal)
title('Cleaned signal')

audiowrite('clean_OSR_us_000_0010_8k.wav', reconstructedSignal, fs);
% In the case of the signal used, there was too much attenuation
```

Gain function

```
function gain = gainFunc(Rprio, Rpost)
theta = (1 + Rpost).*(Rprio./(1 + Rprio));
M = MFunc(theta);
gain = (sqrt(pi)/2).*sqrt((1./(1 + Rpost)).*(Rprio./(1 + Rprio))).*M;
end
```

M function

```
function M = MFunc(theta)
M1 = exp(-0.5*theta);
I0 = besseli(0, (0.5*theta));
I1 = besseli(1, (0.5*theta));
M2 = (1 + theta).*I0 + theta.*I1;
M = M1.*M2;
end
```

Speech detection function

This is a very basic one based on one in an example. In future a much better algorithm could be chosen.

```
function [NoiseFlag, SpeechFlag, NoiseCounter, Dist] = detectSpeech(signal, noise, Noise(
specDistThresh = 3; % default spectral distance threshold
permittedNoiseAmount = 8;

% Calculate distances in spectrogram
specDist= 20*(log10(signal)-log10(noise));
specDist(find(specDist<0)) = 0; % replace negatives with 0

Dist = mean(specDist);
if (Dist < specDistThresh)
    NoiseFlag = 1;
    NoiseCounter = NoiseCounter + 1;
else
    NoiseFlag = 0;
    NoiseCounter = 0;
end

% Detect noise only periods
if (NoiseCounter > permittedNoiseAmount)
    SpeechFlag = 0;
else
    SpeechFlag = 1;
end
```

Sources

Cappe, O. 1994, "Elimination of the musical noise phenomenon with the Ephraim and Malah noise suppressor", IEEE Transactions on Speech and Audio Processing, vol. 2, no. 2, pp. 345-349.

Ephraim, Y. & Malah, D. 1984, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 32, no. 6, pp. 1109-1121.

Marzinzik, M. 2000, Noise reduction schemes for digital hearing aids and their use for the hearing impaired, University of Oldenburg.