转 [投稿]谱减法语音降噪原理

2015年08月04日 14:57:22　阅读数：32687

本文是音频处理的朋友icoolmedia（QQ：314138065）的投稿。各位做视音频技术朋友如果好的原创技术文章并希望通过我的博客分享给大家，也欢迎投稿到我的邮箱：leixiaohua1020@126.com，我会选择内容合适的文章注明作者及联系方式后进行发布。希望通过这种方式帮助大家结交更多的同道中人~

谱减法语音降噪基本原理

谱减算法为最早的语音降噪算法之一，它的提出，基于一个简单的原理：假设语音中的噪声只有加性噪声，只要将带噪语音谱减去噪声谱，就可以得到纯净语音幅度。这么做的前提是噪声信号是平稳的或者缓慢变化的。

得到纯净信号的幅度谱后，可以结合带噪语音相位（近似带替纯净语音相位），从而得到近似的纯净语音，可以这么做的原因是因为语音信号相位不会对语音可懂度造成影响。
按上述所示，如果我们设y(n)为受噪声污染的信号，则y(n)由纯净语音信号x(n)和加性噪声d(n)组成，即：y(n)=X(n)+d(n)。其傅里叶变换后表示为：Y(ω)=X(ω)+D(ω)，或写为：

X(ω) = Y(ω) – D(ω)，如果用功率谱表示可以写为：

$$|Y(\omega)|^2=|X(\omega)|^2+|D(\omega)|^2+X(\omega)\overline{D(\omega)}+\overline{X(\omega)}X(\omega)=|X(\omega)|^2+|D(\omega)|^2+2\mathrm{Re}\{X(\omega)\overline{D(\omega)}\}$$

这里 $2\mathrm{Re}\{X(\omega)\overline{D(\omega)}\}$ 被称为交叉项，我们假定d(n)具有0均值，并且与x(n)不相关，则交叉项为0，上述公式简化为：

$$|Y(\omega)|^2=|X(\omega)|^2+|D(\omega)|^2$$

或写为：

$$|X(\omega)|^2=|Y(\omega)|^2-|D(\omega)|^2$$

音乐噪声和过减因子、谱下限的关系

如果带噪语音的幅度谱（功率谱也同此理）与估计出来的噪声谱相减出现负值时，说明对噪声出现了过估计问题，对这种现象最简单的处理就是将负值设为0，以保证非负的幅度谱。但是对负值的这种处理，会导致信号帧频谱的随机位置上出现小的，独立的峰值。

转换到频域后，这些峰值听起来就像帧与帧之间频率随机变化的多频音，这种情况在清音段尤其明显，这种由于半波整流引起的“噪声”被称为“音乐噪声”。从根本上，通常导致音乐噪声的原因主要有：
（1）对谱减算法中的负数部分进行了非线性处理
（2）对噪声谱的估计不准
（3）抑制函数（增益函数）具有较大的可变性

减小音乐噪声的方法是对噪声谱使用过减技术，同时对谱减后的负值设置一个下限，而不是将它们设为0，其技术形式如下：

$$|\hat{X}(\omega)|^2=|Y(\omega)|^2-\alpha|\hat{D}(\omega)|^2;\text{如果}|Y(\omega)|^2>(\alpha+\beta)|\hat{D}(\omega)|^2$$
$$|\hat{X}(\omega)|^2=\beta|\hat{D}(\omega)|^2$$

其中alpha（大于等于1）为过减因子，它主要影响语音谱的失真程度。Beta（大于0小于1）是谱下限参数，可以控制残留噪声的多少以及音乐噪声的大小。

使用过减因子与谱下限的动机在于：当从带噪语音谱中减去噪声谱估计的时候，频谱中会残留一些隆起的部分或谱峰，有些谱峰是宽带的，有些谱峰很窄，看起来像是频谱上的一个脉冲。通过对噪声谱的过减处理，我们可以减小宽带谱峰的幅度，有时还可以将其完全消除。但是仅仅这样还不够，因为谱峰周围可能还存在较深的谱谷。因此需要使用谱下限来“填充”这些谱谷。

在高信噪比中，alpha应取小值；对低信噪比中，alpha建议取大值。Berouti等人做了大量实验来确定alpha与beta的最优值，在这里我们直接使用就可以了。具体请参考论文：Enhancement of speech corrupted by a acoustic noise。

下面给出谱减算法的Matlab验证代码，调用方法为specsub('filename.wav','outfile.wav');

```plain
function specsub(filename,outfile)
if nargin < 2
   fprintf('Usage: specsub noisyfile.wav outFile.wav \n\n');
   return;
end

[x,fs,nbits] = wavread(filename);
len = floor(20*fs/1000);            % Frame size in samples
```

```
len = floor(20*fs/1000);             % frame size in samples
if rem(len,2) == 1, len=len+1; end;
PERC = 50;                           % window overlap in percent of frame size
len1 = floor(len*PERC/100);
len2 = len-len1;

Thres = 3;      % VAD threshold in dB SNRseg
Expnt = 2.0;    % power exponent
beta = 0.002;
G = 0.9;

win = hamming(len);
winGain = len2/sum(win); % normalization gain for overlap+add with 50% overlap

% Noise magnitude calculations - assuming that the first 5 frames is noise/silence
nFFT = 2*2^nextpow2(len);
noise_mean = zeros(nFFT,1);
j=1;
for k = 1:5
   noise_mean = noise_mean+abs(fft(win.*x(j:j+len-1),nFFT));
   j = j+len;
end
noise_mu = noise_mean/5;

%--- allocate memory and initialize various variables
k = 1;
img = sqrt(-1);
x_old = zeros(len1,1);
Nframes = floor(length(x)/len2)-1;
xfinal = zeros(Nframes*len2,1);

%========================    Start Processing   ==============================
for n = 1:Nframes
    insign = win.*x(k:k+len-1);      % Windowing
    spec = fft(insign,nFFT);         % compute fourier transform of a frame
    sig = abs(spec);                 % compute the magnitude
    %save the noisy phase information
    theta = angle(spec);
    SNRseg = 10*log10(norm(sig,2)^2/norm(noise_mu,2)^2);
    if Expnt == 1.0     % 幅度谱
        alpha = berouti1(SNRseg);
    else
        alpha = berouti(SNRseg); % 功率谱
    end
    %&&&&&&&&&
    sub_speech = sig.^Expnt - alpha*noise_mu.^Expnt;
    diffw = sub_speech - beta*noise_mu.^Expnt;     % 当纯净信号小于噪声信号的功率时
    % beta negative components
    z = find(diffw <0);
    if~isempty(z)
        sub_speech(z) = beta*noise_mu(z).^Expnt;   % 用估计出来的噪声信号表示下限值
    end
    % --- implement a simple VAD detector --------------
    if (SNRseg < Thres)   % Update noise spectrum
        noise_temp = G*noise_mu.^Expnt+(1-G)*sig.^Expnt;    % 平滑处理噪声功率谱
        noise_mu = noise_temp.^(1/Expnt);                   % 新的噪声幅度谱
    end
% flipud函数实现矩阵的上下翻转，是以矩阵的“水平中线”为对称轴
%交换上下对称元素
    sub_speech(nFFT/2+2:nFFT) = flipud(sub_speech(2:nFFT/2));
    x_phase = (sub_speech.^(1/Expnt)).*(cos(theta)+img*(sin(theta)));
    % take the IFFT
    xi = real(ifft(x_phase));
    % --- Overlap and add ---------------
    xfinal(k:k+len2-1)=x_old+xi(1:len1);
    x_old = xi(1+len1:len);
    k = k+len2;
end

wavwrite(winGain*xfinal,fs,16,outfile);

function a = berouti1(SNR)
if SNR >= -5.0 & SNR <= 20
    a = 3-SNR*2/20;
else
    if SNR < -5.0
        a = 4;
    end
    if SNR > 20
        a = 1;
    end
end

function a = berouti(SNR)
if SNR >= -5.0 & SNR <= 20
    a = 4-SNR*3/20;
else
    if SNR < -5.0
        a = 5;
    end
    if SNR > 20
        a = 1;
```

```
100.      end
101.  end
```

几种改进的谱减算法

（1）非线性谱减
Berouti等人提出的谱减算法，假设了噪声对所有的频谱分量都有同等的影响，继而只用了一个过减因子来减去对噪声的过估计。现实世界中的噪声并非如此，这意味着可以用一个频率相关的减法因子来处理不同类型的噪声。

（2）多带谱减法
在多带算法中，将语音频谱划分为N个互不重叠的子带，谱减法在每个子带独立运行。将语音信号分为多个子带信号的过程可以通过在时域使用带通滤波器来进行，或者在频域使用适当的窗。通常会采用后一种办法，因为实现起来有更小的运算量。

多带谱减与非线性谱减的主要区别在于对过减因子的估计。多带算法针对频带估计减法因子，而非线性谱减算法针对每一个频点，导致频点上的信噪比可能有很大变化。这种剧烈变化是谱减法中所遇到的语音失真（音乐噪声）的原因之一。相反，子带信噪比变化则不会特别剧烈。

（3）MMSE谱减算法
上面的方法中，谱减参数alpha和beta通过实验确定，无论如何都不会是最优的选择。MMSE谱减法能够在均方意义下最优地选择谱减参数。具体请参考论文：A parametic formulation of the generalized spectral subtractor method

（4）扩展谱减法
基于自适应维纳滤波与谱减原理的结合。维纳滤波用于估计噪声谱，然后从带噪语音信号中减去该噪声谱。具体请参考以下两篇论文：
Extended Spectral Substraction:Description and Preliminary Results.
Extended Spectral Substraction

（5）自适应增益平均的谱减
谱减法中导致音乐噪声的两个因素在于谱估计的大范围变化以及增益函数的不同。对于第一个问题，Gustafsson等人建议将分析帧划分为更换小的子帧以得到更低分辨率的频谱。子帧频谱通过连续平均以减小频谱的波动。对于第二个问题Gustafsson等人提出使用自适应指数平均，在时间上对增益函数做平滑。此外，为了避免因使用零相位增益函数导致的非因果滤波问题，Gustafsson等人建议在增益函数中引入线性相位。具体请参考论文：Spectral subtraction using reduced delay convolution and adaptive averaging

（6）选择性谱减法
前面提到的方法对所有语音都做同样处理。并不区分是浊音段还是清音段。区分浊音与清音的谱减法有：

1>
双频带谱减法。通过将带噪语音能量与某一阈值进行比较，把语音帧分为浊音和清音。对于浊音帧，用算法确定一个截止频率，在该截止频率之上，语音被认为是随机信号。浊音段则通过滤波分为两个频带，一个频带位于截止频率之下（低通滤波后的语音），另外一个频带高于截止频率（高通滤波后的语音）。然后对低通和高通后的语音信号使用不同的算法进行处理。对低通语音部分在短时傅立叶变换的基础上使用过减算法，对于高通部分以及清音段，使用Thomson的多窗谱估计器取代FFT估计器。主要目的在于减小高频部分的频谱值的波动。具体请参考论文：Adaptive two-band spectral subtraction with multi-window spectral estimation

2>
双激励语音模型法，该算法把语音分为两个独立的组成部分--浊音分量和清音分量。也就是说，语音由这两个分量的和来表示（注意不同于将语音分为浊音段和清音段）。浊音分量的分析是基于对基音频率和谐波幅度的提取。然后从带噪语音谱中减去浊音谱就得到了清音谱。然后使用一个双通道系统，基中一个包括改进的维纳滤波器，被用于增强清音谱。最终增强的语音由增强后的浊音分量和清音分量求和得到。具体请参考论文：Speech enhancement using the dual excitation speech model

3>
还有一种基于浊音、清音的谱减算法，在该算法中语音帧首先根据能量和过零率被划分为浊音和清音。然后将带噪语音谱与锐化函数进行卷积，清音的频谱就会被锐化（用锐化函数进行镨锐化的目的在于增加谱对比度，即在抑制谱谷的同时使谱峰更加突出）。具体请参考论文：Spectral subtraction based on phonetic dependency and masking effects

（7）基于感知特性的谱减
前面提到的方法，谱减参数要么是通过实验计算短时信噪比得到，要么是通过最优均方误差得到，均没有考虑听觉系统的特性，该算法的主要目的是使残余噪声在听觉上难以被察觉。利用了人类听觉系统改进系统的可懂度（即人耳的掩蔽效应）

本文来自于icoolmedia（QQ：314138065）。

文章标签： 音频 算法 Matlab 语音
个人分类： 音频编码