ECE 1018

SIGNAL ANALYSIS AND PROCESSING

PROJECT REPORT

TITLE: SEPARATION OF SINGING VOICE AND MUSIC

PROJECT GUIDE: DR. KALAIVANI S

PROJECT MEMBERS

AMRESHWAR RAO NAIDU	**16BIS0075**
ARUN KUMAR VARMA	**16BIS0096**
AYUSH GUPTA	**16BIS0108**
S KSHIPRA PRASADH	**16BIS0135**

VIT UNIVERSITY VELLORE 632014

CERTIFICATE

Certified that this project “SEPARATION OF SINGING VOICE FROM MUSIC’’ is the bonafide work of AMRESHWAR RAO NAIDU 16BIS0075 ARUN KUMAR VERMA- 16BIS0096, AYUSH GUPTA 16BIS0108, S KSHIPRA PRASADH- 16BIS0135 who carried out the project work under my supervision.

ABSTRACT

Separation of singing voice and music is an interesting research topic since singing voice contains a lot of information, such as melody, singer's characteristic, lyrics, emotion, etc. All of these resources in singing voice are useful for music information retrieval, singer identification, melody extraction, audio content analysis, or even karaoke singing.

However, it is challenging as the methods available right now are not very efficient. Most songs have their own repeating tunes over which the singers sing. This work studies the repeating structure of music and implement the algorithm based on the repeating pattern of the music background.

INTRODUCTION

Our basic algorithm addresses these issues during implementation.

- Repeating period identification: finding the repeating period in mixture.
- Repeating segment modelling: using the repeating period to segment the music into several segments and defining the repeating segment.
- Repeating patterns extraction: using the repeating segment model to further remove the singing voice from the mixture

One significant difference between the singing voice and the music is that the music background always has the repeating pattern, but the singing voice varies with much less repeating pattern. Using this difference between the singing voice and the music, this project provides an algorithm to find the repeating period of music and extract the singing voice from the mixture.

It’s worthy to note that the effect of this algorithm depends significantly on the repeating pattern of the music. The more repeating the background music is, the more effectively this algorithm works.

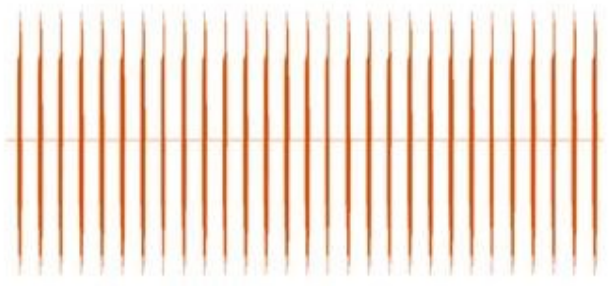
WORKING

Repeating Period Identification

With time interval of 0.04 seconds, 2048 samples and frequency of 44100 HZ, we calculate the Short--Time Fourier transform of mixture signal in MATLAB, we could obtain the mixture spectrogram for the whole song. Using the autocorrelation on mixture spectrogram, that is, comparing the segment and its lagged version over successive time interval to measure the similarity in the segment

After normalization using the first term of beat spectrum b, we could get the final beat spectrum. If a mixture contains the repeating structure, there will be several peaks occur periodically in the beat spectrum. The basic idea is that the time between two peaks that occur periodically in the beat spectrum is the repeating period we need.

←VOICE SIGNAL



← MUSIC SIGNAL

← MIXTURE OF VOICE AND MUSIC

Repeating Segment Modelling

After obtaining the repeating period, we could use the repeating period to evenly time--segment the mixture spectrogram into several segments of length of the repeating period

The rationale is that the since the mixture spectrogram is segmented according to the repeating period, the median of each segments of the mixture spectrogram should be able to capture the repeating pattern of music background and remove the non--repeating singing voice foreground without the impact of outliers

Repeating Pattern Extraction

After getting the repeating segment model, we compare each segment of the mixture spectrogram, which we derived in segmentation, with the repeating segment, which is the median of all segments of mixture spectrogram. The rationale is that if the value of a segment of the mixture spectrogram is bigger than the repeating segment, it denotes that in this segment, it contains more non--repeating information. In order to remove the non--repeating pattern, we need to replace this segment with the repeating segment. Otherwise, if the value of a segment is smaller than the repeating segment, it denotes that this segment contains less non--repeating pattern and we just keep it

We divide repeating spectrogram W by mixture spectrogram V to get the time-frequency mask M. If some parts of the mixture spectrogram are similar to the repeating spectrogram, the value of W / V will be near 1 and these parts is counted as music background with repeating pattern. Otherwise, the value of W / V will be near 0 and these parts will be counted as non--repeating singing voice foreground. The mask M contains the repeating information of the mixture spectrogram and all values in mask M are in the range from 0 to 1. Then, we multiply the mask M with the original mixture spectrogram V. Since the range of all values of M is from 0 to 1, the music part will be reserved after multiplication while the singing voice is removed. That is to say, the result of M * V is the music spectrogram with singing voice removed

Removal of noise from the singing voice

Sometimes, the voice signal may be corrupted with noise during the processing. Hence, we use the LMS filter algorithm for noise removal in the voice signal. Least Mean Square filter (LMS) filter is used to filter the noise by calculating the least mean square of the error signal which is the difference between the actual signal and required signal. LMS filters are simple to implement. It can also be described in the frequency domain. Normalized Least Mean Square algorithm is used for Gaussian noise.

The algorithm uses a linear filter and an adaptive algorithm. The input signal is fed into the linear filter and sent to the feedback system consisting of the adaptive algorithm.

The LMS algorithm can be defined as follows

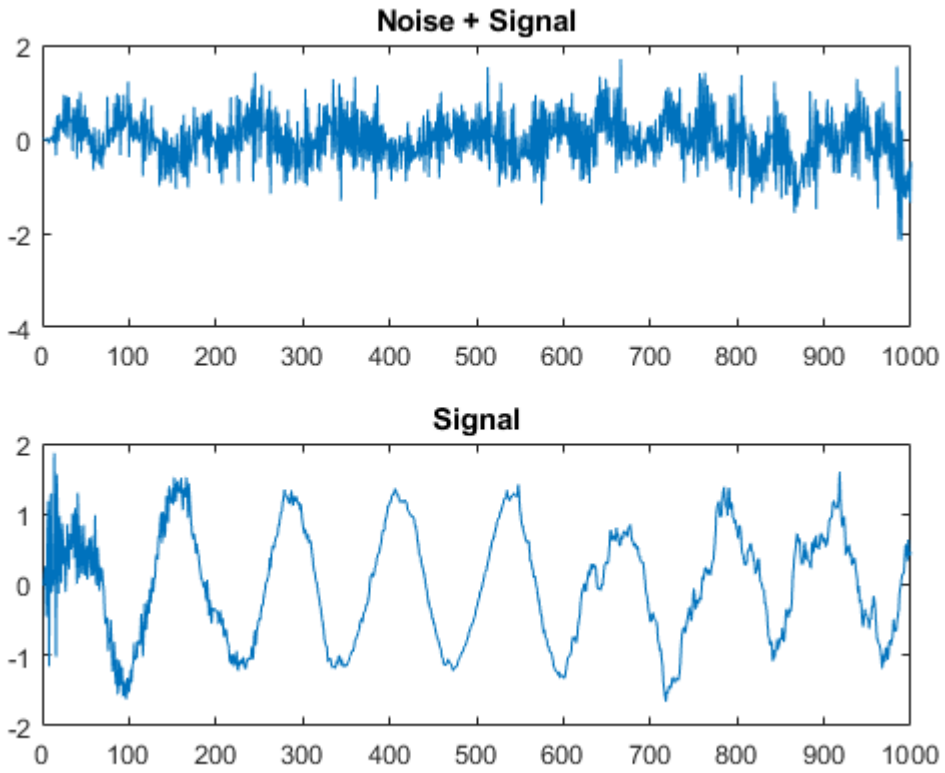
Parameters:	x filter order
	a step size
Initial conditions :	$\hat{h}(0)$=zeros(x)

X(n)=[x(n),x(n-1),x(n-2),….x(n-p+1)]’

LMS adaptive filter also uses the concept of error signal. The error signal e(n) is defined as

e(n)= d(n)-y(n)x(n)

where y(n) is $\hat{h}^H(n)$ and h(n) are the weighted means for the nth signal.



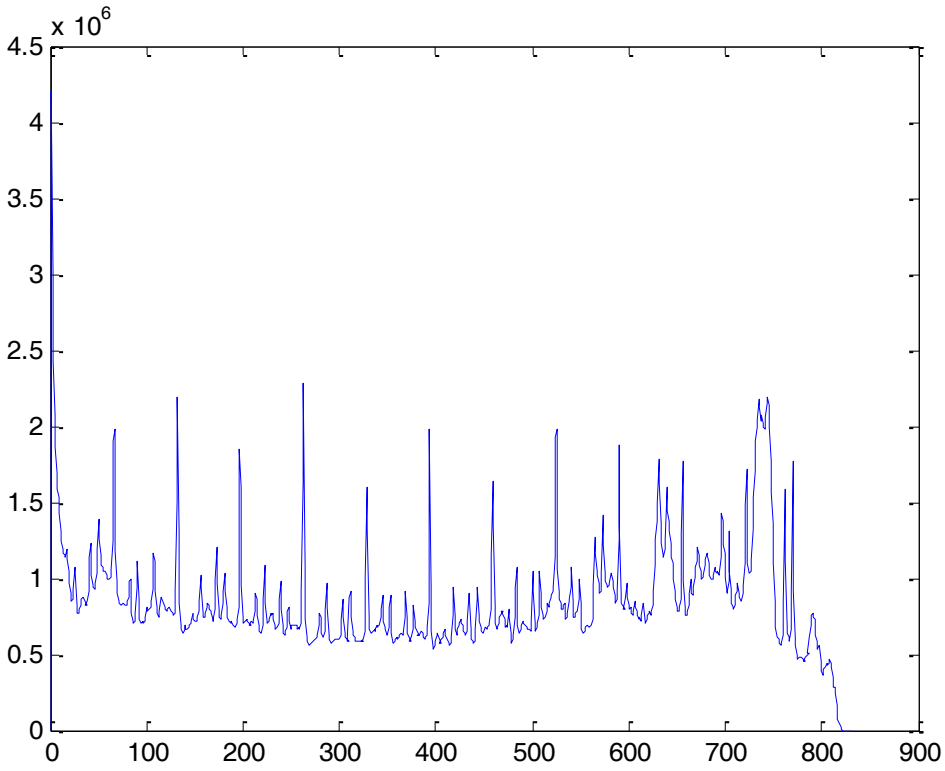
MATLAB CODES

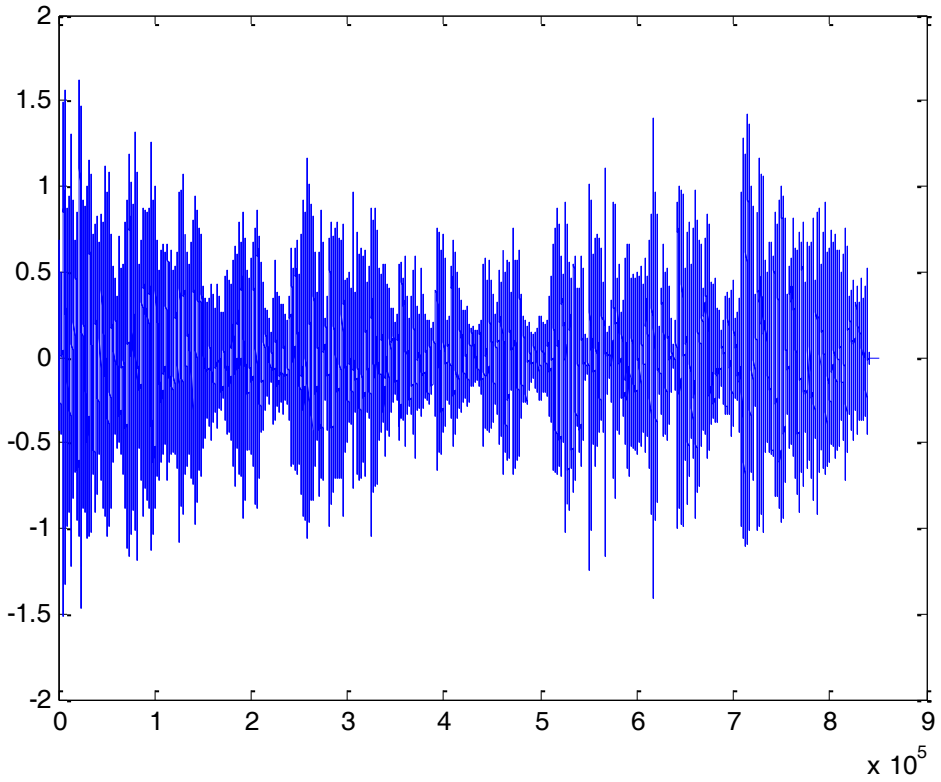
Separate

```
function vocal = separate()
[x, fs] = wavread('song.wav');
len = 0.04;
N = 2 .^ nextpow2(len * fs);
window = hamming(N, 'periodic');
step = N / 2;
[t, k] = size(x);
if (k == 2)
    x = x(:, 1) + x(:, 2);
end
k = 1;
X = stft(x(:, 1), window, step);
V = abs(X(1 : N / 2 + 1, :, :));
VSquare = mean(V .^ 2, 3);
b = beatSpectrum(V);
b(1) = [];
up = min(8, (length(x) / fs) / 3);
```

```
range = [0.8, up];
range = ceil(range * fs / step);
b = b(range(1) : range(2));
[~, p] = max(b);
p = p + range(1) - 1;
y = zeros(t, k);                          % Background music
vocal = zeros(t, k);
Mask = mask(V(:, :, 1), p);
Mask = cat(1, Mask, flipud(Mask(1 : end - 2, :)));
yPart = istft(Mask .* X(:, :, 1), window, step);
y(:, 1) = yPart(1 : t);
vocal = x - y; % Vocal signal;
sound(vocal,44100)
figure(2)
plot(vocal)
end
```

output





Drawspectrogram

```
function x = drawSpectrogram()
[x, fs] = wavread('song.wav');
x = x(:, 1);
window = hamming(2048);
noverlap = 1024;
nttf = 4096
[S, F, T, P] = spectrogram(x, window, noverlap, nttf, fs, 'yaxis');
surf(T, F, 10*log10(P), 'edgecolor', 'none'); axis tight; view(0, 90);
end
```

STFT

```
function X = stft(x, window, step)
[x, fs] = wavread('song.wav');
x = x(:, 1);
window = hamming(512);
step = 512;
t = length(x);
N = length(window);
m = ceil((N - step + t) / step);
x = [zeros(N - step, 1) ; x ; zeros(m * step - t, 1)];
X = zeros(N, m);
for j = 1 : m
    X(:, j) = fft(x((1 : N) + step * (j - 1)) .* window);
end
```

MASK

```
function M = mask(V, p)
% INPUT:
% V: mixture spectrogram
% p: repeating period
[N, m] = size(V);
r = ceil(m / p);                % Number of repeating segments
S = [V, nan(N, r * p - m)];     % Padding
Temp = [];
for i = 1 : N
    for j = 1 : p
        sum = [];
        for k = 0 : r - 2
            sum(k + 1) = S(i, k * p + j);
        end
        Temp(i, j) = median(sum);
    end
end
S = repmat(Temp, [1, r]);       % Get repeating segment S
S = S(:, 1 : m);                % Make S have size [N, m]
W = min(V, S);
M = (W + eps) ./ (V + eps);     % Mask
end
```

ISFT

```
function x = istft(X, window, step)
% INPUT:
% X: signal
% window: analysis window
% step: analysis step
X = X(:, 1);
window = hamming(512);
step = 512;
[N, m] = size(X);                                   % Number of frequency bins
and time frames
l = (m - 1) * step + N;                             % Length with zero-padding
x = zeros(l, 1);
for j = 1 : m                                       % Loop over the frames
    x((1 : N) + step * (j - 1)) = x((1 : N) + step * (j - 1)) +
real(ifft(X(:, j)));   % ifft
end
x(l - (N - step) + 1 : l) = [];                     % Remove zero-padding at
the beginning
x(1 : N - step) = [];                               % Remove zero-padding at
the end
x = x / sum(window(1 : step : N));                  % Normalization
end
```

BEATSPECTRUM

```
function b = beatSpectrum(V)
[x, fs] = wavread('song.wav');
X = stft(x(:, 1), window, step);
V:abs(X(1:end/2+1,:,:));
VSquare = mean(V.^2,3);
[N, m] = size(VSquare);
B = zeros(N, m);                            % Auto-Correlation
for i = 1 : N
```

```
    for j = 1 : m
        sum = 0;
        for k = 1 : m - j + 1
            sum = sum + VSquare(i, k) * VSquare(i, k + j - 1);
        end
        B(i, j) = 1 / (m - j + 1) * sum;
    end
end
b = mean(B);
b = b';
for p = 1 : j
    b(p) = b(p) / b(1);                    % Nomalization
end
figure, plot(b);                           % Plot the beat spectrum
end
```

CONCLUSION

This algorithm depending on identifying the repeating pattern in music to separate the singing voice from the music and this algorithm is highly sensitive to the repeating period of the music. Identifying the accurate repeating pattern is the core of the algorithm. As long as we could obtain the repeating period of a mixture, we could effectively filter the singing voice from the mixture audio. The disadvantage of this algorithm is that this algorithm still assigns some music in separated voice signal due to the reason that only the parts that have highly repeating pattern of music get separated. However, although this algorithm could not get 100% original voice signal, it is a very simple and easy to implement algorithm.