Project Report on

EXTRACTION OF VOICE AND MUSIC FROM AUDIO SIGNAL

Submitted in partial fulfillment of the requirements

of the degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND TELECOMMUNICATION

by

AISHWARYA NAMBIAR (ROLL NO. 02)
JANAK PISHARODY (ROLL NO. 69)
SWAPNIL PHALKE (ROLL NO. 70)
SONALI PARAB (ROLL NO. 71)

Under the guidance of

DR. RAVINDRA CHAUDHARI

Department of Electronics and Telecommunication Engineering

St. Francis Institute of Technology, Mumbai
University of Mumbai
(2017-2018)

CERTIFICATE

This is to certify that the project entitled **"Extraction of Voice and Music from Audio Signal"** is a bonafide work of **"Aishwarya Nambiar (02), Janak Pisharody (69), Swapnil Phalke (70), Sonali Parab (71)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Electronics and Telecommunication Engineering**.

(Dr. Ravindra Chaudhari)
Project Guide

(Dr. Gautam Shah)
Head of Department

(Dr. Sincy George)
Principal

Project Report Approval for B.E.

This project entitled ***'Extraction of Voice and Music from Audio Signal'*** by ***Aishwarya Nambiar, Janak Pisharody, Swapnil Phalke and Sonali***

Parab is approved for the degree of ***Bachelor of Engineering in Electronics and Telecommunication*** from University of Mumbai.

Examiners:

`

Date:

Place:

ABSTRACT

A music signal is, a mixture of many component signals obtained from various instruments and the vocals. This mixture signal can be broadly classified based on how they lie in the complete signal span as low rank components(instrumental) and sparse components(vocal). The problem tackled in this project is, how can these two components be separated efficiently using some of the available techniques which are discussed. One such technique, Robust Principal Component Analysis, is executed here. This technique involves constraint optimization of a convex function that is given by RPCA definition. In this process, first the STFT of the audio vector is taken and then the matrix is broken down into two vectors and a low rank component using SVD(Singular Value Decomposition). The goal is to minimize the matrix and for this, we use an algorithm called Augmented Lagrangian Method(ALM). The output of the algorithm is the Low-rank Component and sparse component. The obtained result when converted to time domain, we obtain 2 audio files. The quality of the audio file based on the energy ratios can be obtained in the form of parameters like SDR, SIR, SAR where the source represents the individual ground truth, which forms the dataset. the dataset used here is MIR-1K dataset.

Keywords: List of words closely connected with the project. Music, instruments, vocal, STFT, ALM, audio separation, low-rank, sparse, SVD, BSS, evaluation.

Contents

List of Figures

List of Abbreviations

AAC	Advanced Audio Coding
ADC	Analog to Digital Converter
AIFF	Audio Interchange File Format
ALM	Augmented Lagrangian Multiplier
APG	Accelerated Proximal Gradient
BSS	Blind Source Separation
CD	Compact Disc
DFT	Discrete Fourier Transform
FLAC	Free Lossless Audio Codec
IALM	Inexact Augmented Lagrangian Multiplier
IBM	International Business Machines Corporation
MP3	MPEG-1 or MPEG-2 Audio Layer III
PCA	Principal Component Analysis
PCP	Principal Component Pursuit
REPET	REpeating Pattern Extraction Technique
RPCA	Robust Principal Component Analysis
SAR	Signal to Artifacts Ratio
SDR	Signal to Distortion Ratio
SIR	Signal to Interference Ratio
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Transform
SVD	Singular Value Decomposition
SUMT	Sequential Unconstrained Minimization Techniques
WAV	Waveform Audio File Format

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered

to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in this submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Aishwarya Nambiar)

(Janak Pisharody)

(Swapnil Phalke)

(Sonali Parab)

Date:

Chapter 1

Introduction

With respect to a song, a singing voice provides quite useful information since it encloses the lyrics, emotion as well as the singer of the song. If the singing voice is separated from its music it could be of use in many areas such as automatic lyrics recognition and alignment, singer identification, and music information retrieval. However in these applications music accompaniment seems to act as a form of noise or interference to the impending vocal part. If

humans are to be considered then separating these parts is quite easy since we have in built human auditory system. But the same thing to be implemented in machines is difficult.

Describing the instrumental/musical part, it contains a wide range of sound components. According to the components, the frequency can be anywhere between 32 Hz to 7902 Hz, with different ranges for different instruments playing in the music. The chart shown in figure 1.1 represents the frequency ranges of different instruments[1]. The orange color represents the fundamental frequencies of each specific instrument while the yellows signify their harmonics. Low fundamentals are the blacks on the left while the black surrounded by yellow represents air.

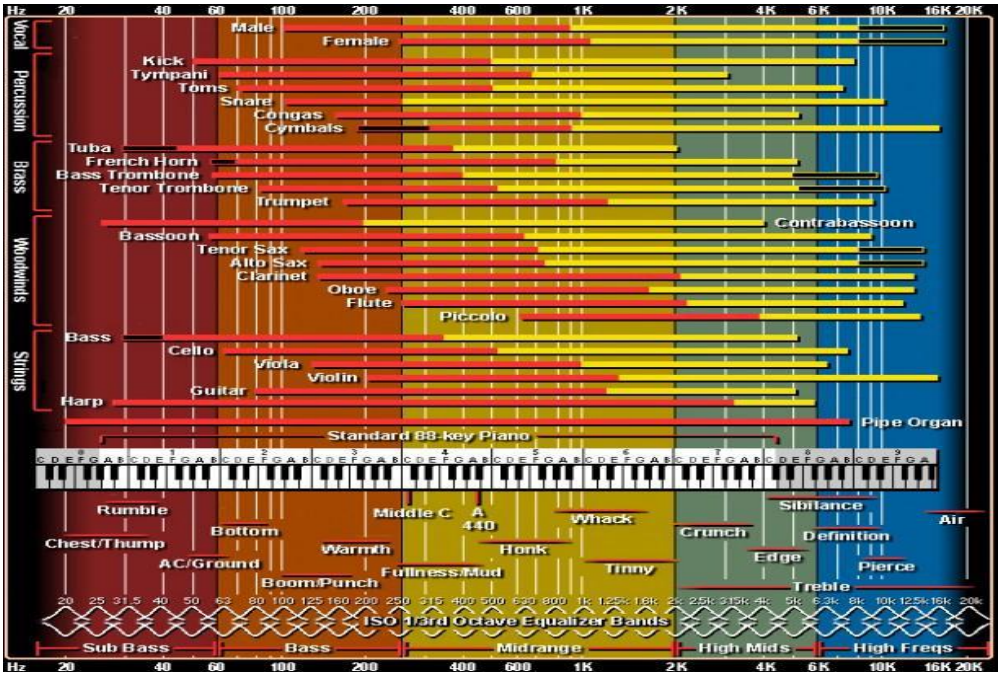


Figure 1.1: Equalizer Chart

Previous work on singing-voice separation systems are classified into two categories:

1] Supervised systems - It includes mapping signals onto a feature space, then detecting singing voice segments. In the end applying source separation techniques such as non-negative matrix factorization, adaptive Bayesian modeling, and pitch-based interference.

2] Unsupervised systems - In these systems no prior training or particular features, such as the source/filter model and the autocorrelation-based method

1.1 Motivation

The recent explosion of massive amounts of high-dimensional data in science, engineering, and society presents a challenge as well as an opportunity to many areas such as image, video, multimedia processing, web relevancy data analysis, search, biomedical imaging and

bioinformatics. In such application domains, data now routinely lie in thousands or even billions of dimensions, with a number of samples sometimes of the same order of magnitude. The ability to efficiently separate a song into its music and voice components would be of great interest for a wide range of applications, among others instrument/vocalist identification, pitch/melody extraction, audio post processing, and karaoke gaming. The techniques for this purpose are limited due to which there is a requirement of music recreation, sometimes from scratch. By developing an efficient algorithm which can separate the vocal and instrumental, it is possible to edit a created song and use it multiple times for different artists and purposes, including changing the emotion and also adding more instrumental components to enhance the quality of music.

The aim of the program will also be to maintain or enhance the quality to make it sufficient to be processed.

1.2 Problem Statement

To separate the sparse and Low-rank component efficiently, i.e. efficient as per subjective assessment as well as better comparison results of energy ratios of source (ground truth) v/s separation obtained by execution of algorithm.

1.4 Organization of Project Report

- **Literature Survey:** Initially a broad survey on the past work and different available techniques is done. Based on the comparison of the results and the quality of output. Out of all the techniques, RPCA was found to be more effective.
- **Study of input signal and Fourier Transform:** A study of music as a signal and its components based on the frequency parameters is done. The study of its Short-Time-Fourier-Transform is done.
- **Study of ALM:** Deciding a coding algorithm based on ALM.
- **Simulation in MATLAB:** Simulation of code in MATLAB.
- **Analyzing the results:** The output signals are analyzed and the flaws of separation is noted.
- **Simulation based on various inputs:** The outputs using different samples of inputs are checked with different lengths, bit rates and sampling frequencies.

Chapter 2

Literature Review

Extraction of music and voice from a mix signal has been an interest of research in recent years, and many learning algorithms and variations have been developed. This can be done by separating the frequency components from the main music signal.

The available techniques are:

1. Robust Principal Component Analysis (RPCA).
2. Repeating Pattern Extraction Technique (REPET).

2.1 Robust Principal Component Analysis (RPCA)

RPCA is a matrix factorization algorithm for solving underlying low-rank and sparse matrices. Our area of interest is to separate vocal and instrumental component from the song. Music accompaniment can be assumed to be low-rank subspace, because of its repeating nature. On the other hand, the singing voice has more variations and is relatively sparse within the song. Based on these assumptions, we use Robust Principal Component Analysis, which is matrix factorization algorithm for solving underlying low-rank and sparse matrices[2].

RPCA is a convex program for recovering low-rank matrices when a fraction of their entries have been corrupted by errors, i.e., when matrix is sufficiently sparse. The approach, Principal Component Pursuit, suggest solving the following convex optimization problem which is shown in equation 2.1,

$$\text{minimize } \|\square\|_* + \square\|\square\|_1$$

...(2.1)

$$\text{subject to } \square + \square = \square$$

where $\square \in \square^{\square_1 \times \square_2}$, $\square \in \square^{\square_1 \times \square_2}$, $\square \in \square^{\square_1 \times \square_2}$, $\|.\|_*$ and $\|.\|_1$ denote the nuclear norm (sum of singular values) and the L-1 norm (sum of absolute values of matrix entries), respectively. $\square > 0$ is trade-off parameter between the rank of $\square$ and the sparsity of $\square$. As suggested in [6], using a value of $\square = \frac{1}{\sqrt{\square\square\square(\square_1,\square_2)}}$ is a good rule of thumb which can then be adjusted slightly to obtain best possible result. By RPCA, we expect the low-rank matrix $\square$ to contain music accompaniment and the sparse matrix $\square$ to contain vocal signals. The separation process is performed according to the block diagram mentioned in fig. 2.1. First, we compute spectrogram of music signal as matrix $\square$, calculated from Short Time Fourier Transform (STFT). Then, we use inexact Augmented Lagrangian Multiplier (ALM) method [4] to solve $\square + \square = |\square|$, given the input magnitude of $\square$. Then by RPCA, we can obtain two output matrices, $\square$ and $\square$.

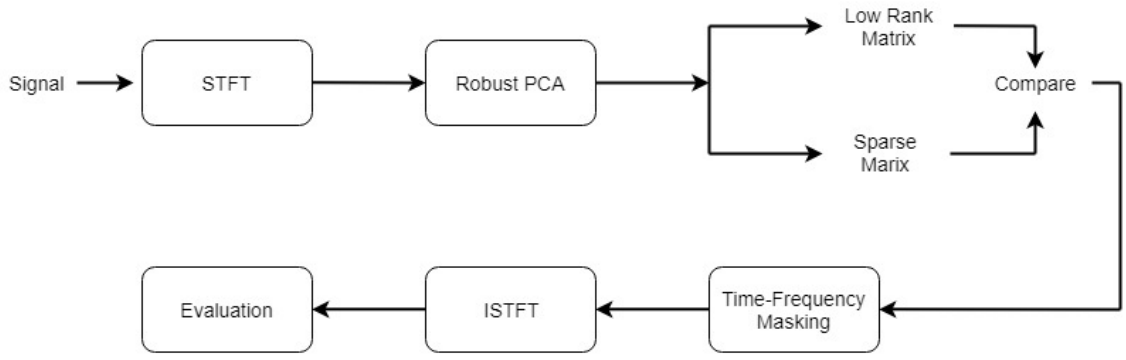


Figure 2.1: Block diagram of RPCA

2.2 Repeating Pattern Extraction Technique (REPET)

Repeating Pattern Extraction Technique considers the signal to be composed of repeating background and non-repeating foreground[5]. Vocal part of the signal, which is more variable in nature, is considered to be the foreground and instrumental part, which is more stable in nature, is considered to be the background. Basically, REPET is an approach for separating background and foreground. The simple procedure REPET follows is:

- Find repeating element in mixture
- Derive underlying repeating model
- Extract repeating background by comparing models to the mixture

TYPES OF REPET:

2.2.1 Original REPET

Original REPET is used when the repeating background is stable or for periodic structure[5]. It consists of three stages:

Step 1: Repeating Period Identification

Given a mixture signal x, first calculate its Short Time Fourier Transform (STFT) X, using Hamming window of w samples. Derive the magnitude spectrogram V by taking the absolute values of the elements of X, after discarding the symmetric part, while keeping the DC components. Compute the autocorrelation of each row of the power spectrogram V^2 (element-wise square of V) and obtain the matrix B as shown in equation 2.2. We use V^2 to emphasize peaks of periodicity in V. The overall acoustic self-similarity b of B is obtained by taking the mean over the rows of B as shown in eq 2.3

Once the beat spectrum is calculated, the first term which measures the similarity of the whole signal with itself (lag 0) is discarded as shown in equation 2.4.

$$B(j,k) = \frac{1}{w-k+1}\sum_{j=0}^{w-k+1} V(j,j)^2 V(j,j+k-1)^2 \quad \ldots(2.2)$$

$$b(k) = \frac{1}{w}\sum_{j=1}^{w} B(j,k) \quad \ldots(2.3)$$

$$b(k) = \frac{b(k)}{b(1)} \quad \ldots(2.4)$$

for $j = 1 \ldots m$ where $m = \frac{n}{2} + 1$ = Number of frequency channels

for $i = 1 \ldots n$ where n = Number of time frames

If periodically repeating patterns are present in V, B would form peaks that are periodically repeating at different period rates. Simple algorithm for finding period p is as follows:

1. For each possible period p in B, check if its integer multiples p (i.e., $p, 2p, 3p$, etc.) correspond to the highest peaks in their respective neighborhoods $[p - \Delta, p + \Delta]$, where Δ is a variable distance parameter, function of p and its value is set to $[3p/4]$.
2. If they do, add their values, minus the mean of the given neighborhood to filter any possible noisy background.
3. Divide the sum by total number of integer multiples of p, leading to mean energy value for each period p.
4. The repeating period p is the period p that gives the largest mean value.

This helps to find the period of the strongest repeating peaks in B, corresponding to the period of the underlying repeating structure in V as exemplified in fig. 2.2.

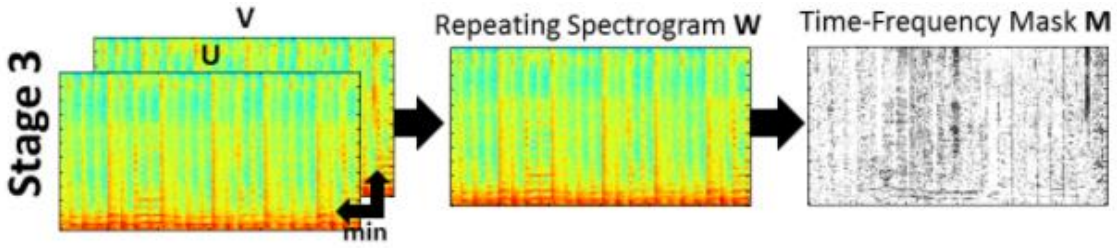


Figure 2.2: Repeating Period Identification (Original REPET)

Step 2: Repeating Segment Modeling

We use period p estimated in previous step to evenly time-segment the spectrogram V into r segments of length p. We define the repeating segment model S as the element-wise median of the r segments. S is calculated as shown in equation 2.5

$$S(i, j) = \text{median}_{k=1 \ldots r}\{V(i, j + (k-1)p)\} \quad \ldots(2.5)$$

for $i = 1 \ldots p$ (frequency) and $j = 1 \ldots p$ (time)

where p = period length and r = number of segments

Time-frequency bins with small deviation at period rate p would most likely represent repeating elements and would be captured by the median model. On the other hand, time-

frequency bins with large deviation at period rate 𝑝 would most likely be corrupted by non-repeating elements and would be removed by the median model, as illustrated in fig. 2.3.

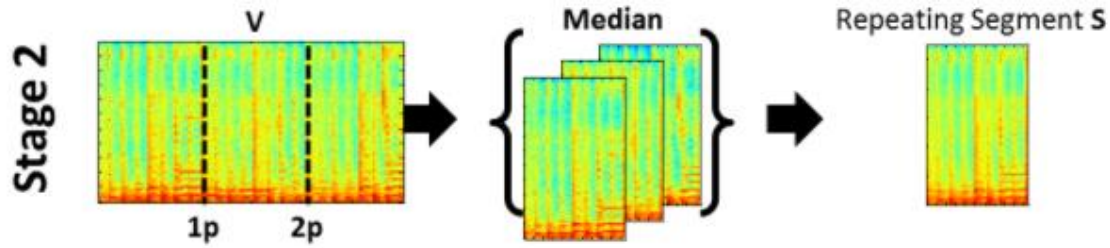


Figure 2.3: Repeating Segment Modeling (Original REPET)

Step 3: Repeating patterns Extraction

Once the repeating segment model 𝑆 is calculated, we use it to derive a repeating spectrogram model 𝑊, by taking element-wise minimum between 𝑆 and each of the 𝑝 segments of the spectrogram 𝑉, as shown in eq 2.2.5.

$$W(f, t + (j-1)p) = \min\{S(f, t), V(f, t + (j-1)p)\} \quad \ldots(2.6)$$

for $f = 1 \ldots m$, $t = 1 \ldots p$, and $j = 1 \ldots q$

The idea is that, if we assume that the non-negative mixture spectrogram 𝑉 is the sum of a non-negative repeating spectrogram model 𝑊 and a non-negative non-repeating spectrogram 𝑉 − 𝑊, then we must have 𝑊 ≤ 𝑉, element-wise. Once the repeating spectrogram 𝑊 is calculated, we use it to derive soft time-frequency mask 𝑀, by normalizing 𝑊 by the mixture spectrogram 𝑉, element-wise as shown in equation 2.7, .

$$M(f, t) = \frac{W(f,t)}{V(f,t)} \quad \ldots(2.7)$$

for $f = 1 \ldots m$ (frequency) and $t = 1 \ldots n$ (time) with $M(f, t) \in [0,1]$

The idea is that time-frequency bins that are likely to repeat at period 𝑝 in 𝑉will have values near *1* in 𝑀 and will be weighted towards repeating background, and time-frequency bins that are not likely to repeat at period 𝑝 in 𝑉 would have values near *0* in 𝑀 and would be weighted toward the non-repeating foreground, as described in fig. 2.4. The time-frequency mask 𝑀 is then symmetrized and applied to the STFT 𝑋 of the mixture 𝑉. The estimated music signal is obtained by converting the resulting STFT into the time domain. The estimated voice signal is obtained by simply subtracting the time-domain music signal from the mixture signal 𝑥.

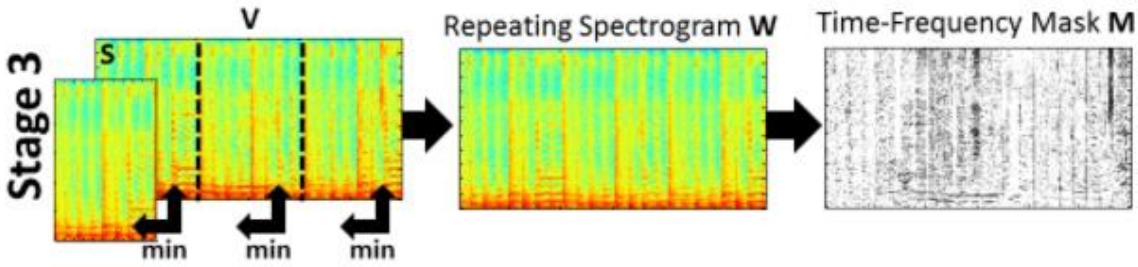


Figure 2.4: Repeating patterns Extraction (Original REPET)

2.2.2 Adaptive REPET

Adaptive REPET[6] overcomes the drawbacks of original REPET. Adaptive REPET analyses each time frame of the signal in order to find the repeating component. Hence, it can extract varying repeating, which is not possible in case of original REPET.

Adaptive REPET follows exact same procedure as that of original REPET. The only difference is, it repeats the procedure for every frame.

Step 1: Repeating Period Identification

The beat spectrum helps to find the global periodicity in a signal. Local periodicities can be found by computing beat spectra over successive windows. Given a mixture signal x, we first compute its magnitude spectrogram V as shown in eq. 2.8 Given a window size $w \leq m$, where m is number of time frames in x, we then compute beat spectrum b_j for every time frame j in x as shown in eq 2.10 using autocorrelation function r_j calculated in eq 2.9. We then concatenate the b_j's into matrix of beat spectra B as shown in eq 2.11.

$$b_j(i,j) = V(i, j + w - \left[\frac{w+1}{2}\right]) \quad \text{...(2.8)}$$

$$r_j(i,j) = \frac{1}{w-j+1}\sum_{k=1}^{w-j+1} V_j(i,j)^2 V_j(i,j+w-1)^2 \quad \text{…(2.9)}$$

$$b_j(j) = \frac{1}{w}\sum_{k=1}^{w} r_j(i,j) \quad \text{…(2.10)}$$

$$B(i,j) = b_j(j) \quad \text{...(2.11)}$$

for $i = 1 \ldots n$ where $n = \frac{w}{2} + 1 =$ Number of frequency channels

for $j = 1 \ldots m$ where $w =$window size

for $j = 1 \ldots m$ and $j = 1 \ldots m$ where m = number of time frames

Once the beat spectrogram B is calculated, the first row (i.e. time lag 0) is discarded. If periodically repeating patterns are present in V, B would form horizontal lines that are periodically repeating vertically, unveiling underlying periodically repeating structure of the mixture. If variations of periodicity happen over time in V, the horizontal lines in B would show variations in their vertical periodicity. We then use a period finder to estimate for every time frame j, the repeating period p_j from the beat spectrum b_j in B, as described in fig. 2.5.

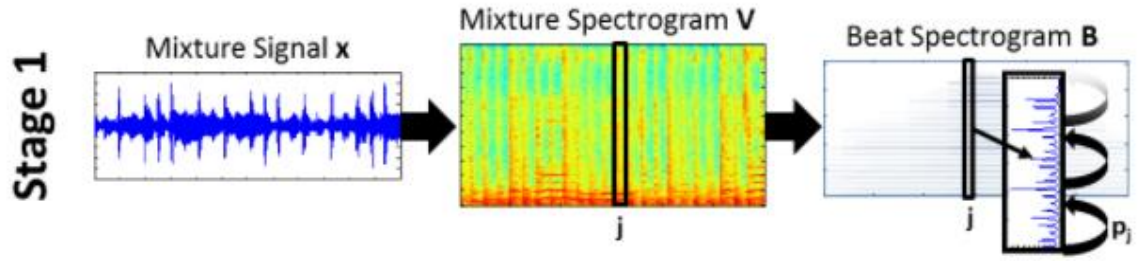


Figure 2.5: Repeating Period Identification (Adaptive REPET)

Step 2: Repeating Spectrogram Modeling

For every time frame j in mixture spectrogram V, we derive the corresponding frame j in W by taking for every frequency channel, the median of the h frames repeating at period rate p_j around j, where h is the maximum number of repeating frames. h is calculated as follows,

$$W(i,j) = \text{median}_{k=1 \ldots h}\{V(i, j + (k - [\tfrac{h}{2}])p_j)\} \quad \ldots \quad (2.12)$$

For $j = 1 \ldots m$ and for $j = 1 \ldots m$

Where h =maximum number of repeating frames

Where p_j = period length for frame j.

The rationale is that, if we assume that the non-repeating foreground has a sparse and varied time-frequency representation compared with the time-frequency representation of the repeating background, time-frequency bins with small deviation at their period rate p_j would most likely represent repeating elements and would be captured by median model.

On the other hand, time-frequency bins with large deviation would be removed by the median mode, as exemplified in fig. 2.6.

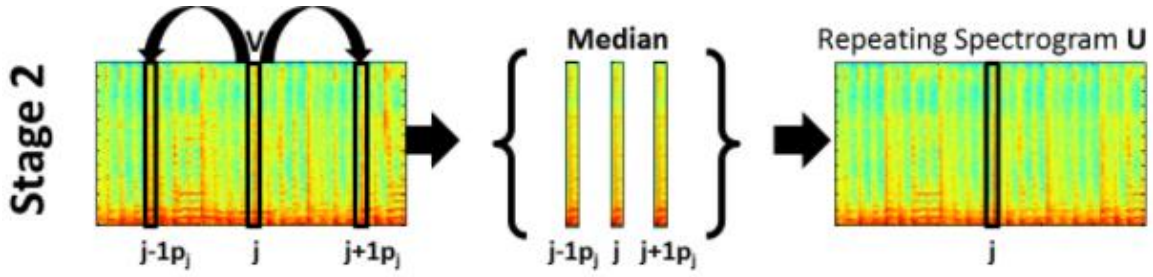


Figure 2.6: Repeating Spectrogram Modeling (Adaptive REPET)

Step 3: Repeating Structure Extraction

We use repeating spectrogram model □ to derive a repeating spectrogram model □, by taking element-wise minimum between □ and the mixture spectrogram □, as shown in eq 2.13

$$□(□, □) = □□□\{□(□, □), □(□, □)\} \quad \text{... (2.13)}$$

for $□ = 1 \ldots □$ and $□ = 1 \ldots □$

Once the repeating spectrogram model □ is calculated, we use it to derive a soft time-frequency mask □, as illustrated in fig. 2.7.

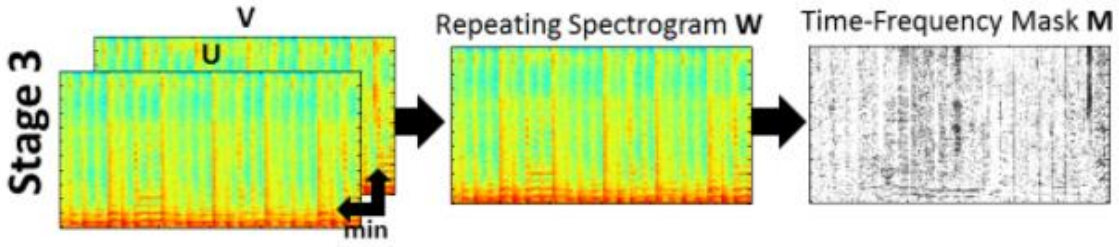


Figure 2.7: Repeating Structure Extraction (Adaptive REPET)

2.2.3 REPET SIM

REPET-SIM uses a similarity matrix to separate the repeating background (instrumental) from the non-repeating foreground (vocal) in a mixture. It can be used to handle non-periodically as well as periodically repeating structures[7][8].

Step 1: Repeating Elements Identification

Repeating/similar elements in a signal can be found by using the similarity matrix, which is a two-dimensional representation where each point (□, □) measures the similarity between any two elements □ and □ of a given sequence. Given a mixture signal □, we first compute its magnitude spectrogram □. We then compute the similarity matrix □ by multiplying transposed □ and □, after normalization of the columns of □ by their Euclidean norm, as shown in eq 2.14

. In other words, each point $(□_□, □_□)$ in □ measures the cosine similarity between the time frames $□_□$ and $□_□$ of □.

$$□(□_□, □_□) = \frac{\sum_{□=1}^{□} □(□,□_□)□(□,□_□)}{\sqrt{\sum_{□=1}^{□} □(□,□_□)^2}\sqrt{\sum_{□=1}^{□} □(□,□_□)^2}} \quad \text{... (2.14)}$$

where $□ = \frac{□}{2} + 1$ = number of frequency channels

for $□_□ = 1 \ldots □$ and $□_□ = 1 \ldots □$

where □ = number of time frames

Once the similarity matrix □ is calculated, we use it to identify the repeating elements in the mixture spectrogram □. If repeating elements are present in □, □ would for regions of high and low similarity at different times, unveiling the underlying repeating structure of the mixture.

We then identify for every time frame □ in □, the frames $□_□$'s that are the most similar to frame □ and save them in a vector of indices $□_□$, as illustrated in fig. 2.8.

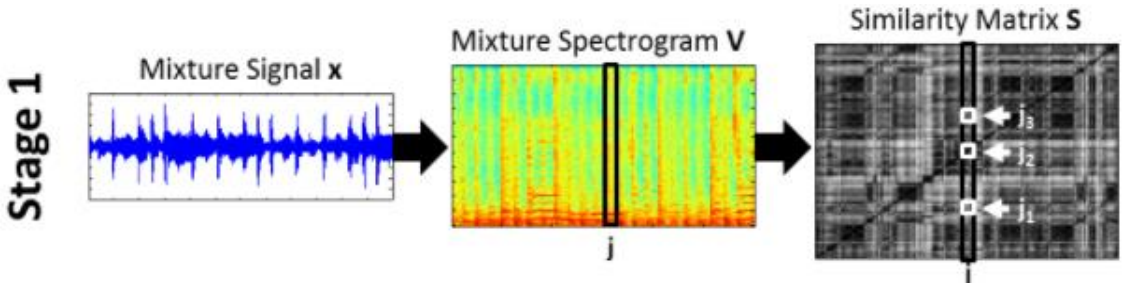


Figure 2.8: Repeating Elements Identification (REPET-SIM)

Step 2: Repeating Spectrogram Modeling

For every time frame □ in the mixture spectrogram □, we derive the corresponding time frame □ in □ by taking for every frequency channel, the median of the repeating frames $□_□$'s given by the vector of indices $□_□$, as shown below,

$$□(□, □) = □□□□□□□ □ = 1 \ldots □ \{□(□, □_□(□))\} \quad \text{... (2.15)}$$

where $□_□ = □_1 \ldots □_□$ =indices of repeating frames

where □ =maximum number of repeating frames

for $□ = 1 \ldots □$ and for $□ = 1 \ldots □$

Assuming that the non-repeating foreground has a sparse and varied time–frequency representation compared with the time–frequency representation of the repeating background, time–frequency bins with small deviations within their repeating frames $□_□$'s would most

likely represent repeating elements and they would be captured by the median model. Whereas the time frequency bins with large deviations within their repeating frames ⍰⍰'s would most likely be corrupted by non-repeating elements (i.e., outliers) and would be removed by the median model, as described in fig. 2.9.

REPET-SIM also looks for non-periodically repeating elements for each time frame, so that it can also handle non-periodically repeating structures where repeating elements can also happen intermittently.

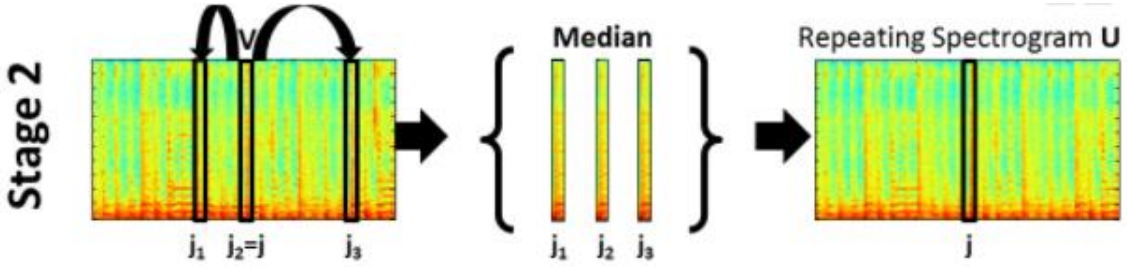


Figure 2.9: Repeating Spectrogram Modeling (REPET-SIM)

Step 3: Repeating Structure Extraction

After calculating the initial repeating spectrogram model ⍰, a refined repeating spectrogram model ⍰ is derived from this initial spectrogram model. Later this repeating spectrogram model ⍰ is used to derive a soft time-frequency mask ⍰, as exemplified in fig. 2.10.

REPET-SIM can be used to implement real-time computing online, especially for real-time speech enhancement. The Online REPET-SIM processes the time frames consecutively and temporarily stores the past frames in a buffer. For every time frame being processed, the similarities calculated with the past frames stored in the buffer. The median is then taken between the frame being processed and its most similar frames for every frequency channel, leading to the corresponding time frame for the repeating background.

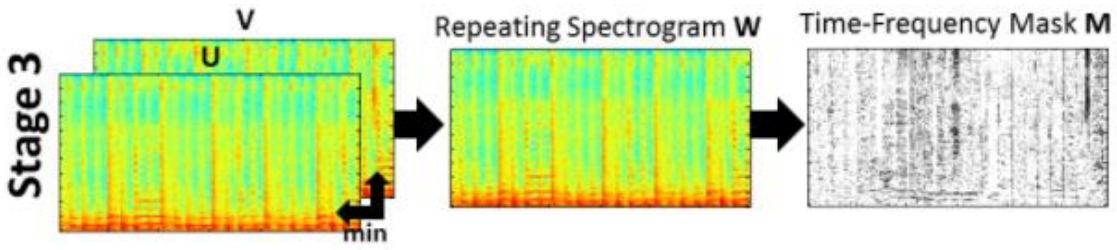


Figure 2.10: Repeating Structure Extraction (REPET-SIM)

Chapter 3

Audio Formats

3.1 WAV format:

Waveform Audio File format is the most basic audio format. It is a raw audio format which was developed jointly by IBM and Microsoft. It stores audio data in blocks. WAV files are usually stored uncompressed, which means that they can get quite large, but they cannot exceed 4 gigabytes due to the fact that the file size header field is a 32-bit unsigned integer (32 bit file length means a maximum of 4 gigs). WAV files are stored in a binary format. They are comprised of chunks, where each chunk tells you something about the data in the file.

They work by taking an audio signal and converting it to binary data. To do this, a device called an analogue to digital converter (ADC) takes snapshot 'slices' thousands of times per second. For example, CD quality audio records at 44.1kHz, meaning it records at 44,100 slices per second. This makes it capable of recording the entire audible frequency range of 20hz-20khz.

Advantages of WAV format:

- It is an accurate, lossless format – in a nutshell, this means that the format reproduced the recording accurately without losing audio quality due to the format itself.
- It is a very simple format – as a result of the files simplicity, files are relatively easy to process and edit. This has meant that easy to use editor software is available at all levels (from freeware to full, pro applications)
- Nowadays, astonishingly high recording rates can be achieved, with huge dynamic ranges (many 'home' audio interfaces offer up to 192kHz)

Drawbacks of WAV format:

- File size – WAV files are large. A stereo, CD quality recording (44.1khz, 16-bit), works out at 10.09 MB per minute. Moving up to 48kHz 24-bit stereo (which will improve both the frequency range and the available dynamic range) will increase file size to 16.48 MB per minute.
- Large file size makes WAVs impractical for portable devices and streaming.

For the song "Cleopatra In New York – Karuan Remix" imported into Adobe Audition difference in qualities of music formats are shown

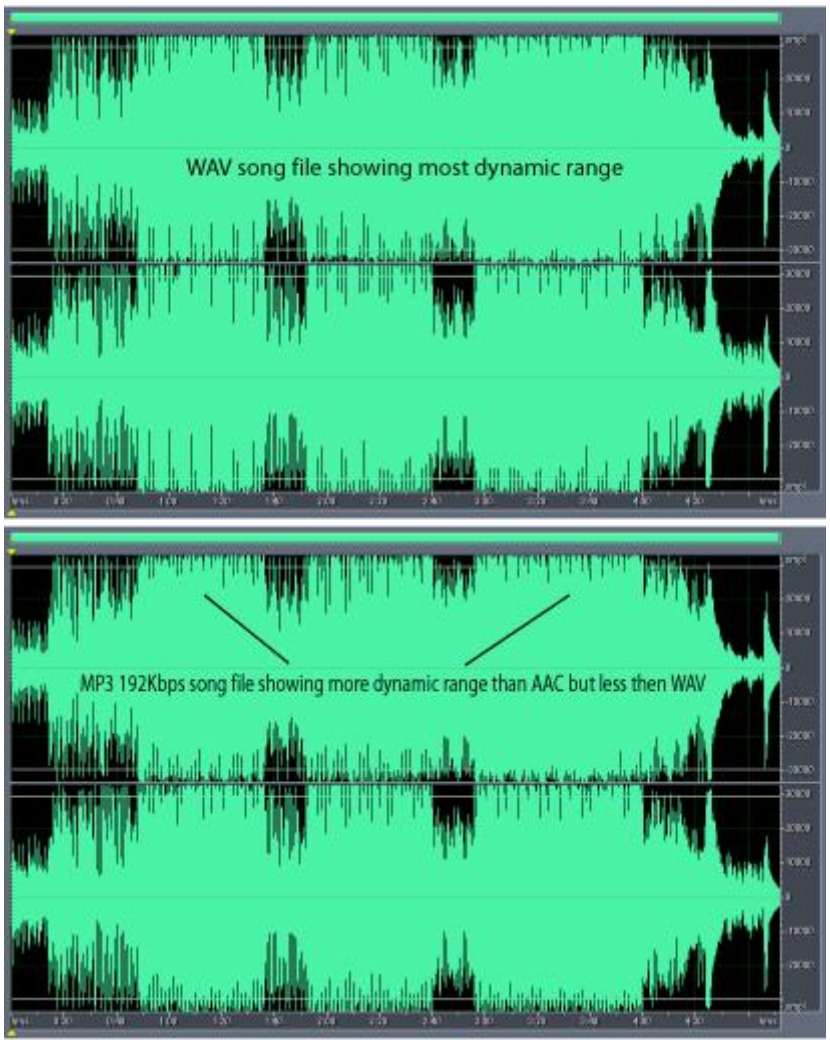


Figure 3.1: Difference in quality for various music formats

3.2 MP3 format:

MPEG-1 Audio Layer-3 is a standard technology and format for compressing a sound sequence into a very small file (about one-twelfth the size of the original file) while preserving the original level of sound quality when it is played. Unlike WAV files, MP3s are a lossy format. This means

that encoding audio to MP3 will reduce its quality, but also reduce its file size.Psychoacoustics is the study of how people perceive sound. There are many anomalies in the way we hear. It is these quirks in our hearing that are exploited by MP3 files.

Firstly, the way in which an MP3 encodes knows that there will be certain frequencies that ear will not hear, and so it gets rid of this data. Secondly, it knows that if there are sounds that are loud, they will mask certain quieter sounds, so it gets rid of the 'hidden' sound. Finally, it exploits the fact that ear will hear some frequencies better than others. It's all tied in with the 'threshold of audibility'.

Once the MP3 has decided which data to keep, this is then compressed again with more traditional data compression techniques. This gets the files down to a fraction of the size of WAV files- usually around 1/10th, but at the expense of some quality.

Advantages of MP3 format:

- Small file format – Because files are so small, they can be easily distributed over the Internet, and huge libraries stored on computers or handheld devices. Because of this, they have become the consumer standard for purchasing music.
- Compresses files with little perceivable difference to the overall sound quality.

Drawbacks of MP3 format:

- To compress the file, audio quality is sacrificed – though it's very clever, MP3 encoding is not perfect. Compression can sometimes result in odd audio 'artefacts' that are detrimental to audio quality, particularly to higher frequencies.
- This audio inaccuracy means that MP3s are not suitable for pro audio work.

3.3 AIFF Format:

Also developed by Apple, the Audio Interchange File Format is an audio file format standard used for storing sound data for personal computers and other electronic audio devices on Apple Macintosh computer systems. The audio data in most AIFF files is uncompressed pulse-code modulation (PCM). AIFF is both an uncompressed (there is also a compressed variant) and lossless audio format. Like Microsoft's WAV file format, AIFF files can take up a lot of digital storage space, making it best for archiving and editing.

The extension for this file type is ".aif" when it is used on a PC. On a Mac, the file extension is not needed.This type of AIFF file uses much more disk space than lossy formats like MP3

An AIFF file contains the raw audio data, channel information (monophonic or stereophonic), bit depth, sample rate, and application-specific data areas.

3.4 OGG format:

Ogg is a free, open container format created under unrestricted software patents by the Xiph.Org Foundation (Ogg Vorbis). The Ogg container format can multiplex a number of independent streams for audio, video, text (such as subtitles), and metadata. An "ogg" file refers to ogg-vorbis, a lossy audio codec. But an ogg file may contain an audio stream compressed with FLAC (lossless). The Ogg container format can multiplex a number of independent streams for audio, video, text (such as subtitles), and metadata. The "Ogg" bitstream format, spearheaded by the Xiph.Org Foundation, has been created as the framework of a larger initiative aimed at developing a set of components for the coding and decoding of multimedia content, which are available free of charge and freely re-implementable in software.

3.5 FLAC format:

It is an audio format similar to MP3 but lossless. In Free Lossless Audio Codec, audio is compressed without any loss in quality. It is the fastest and most widely supported lossless audio codec. FLAC is non-proprietary, is unencumbered by patents, has an open-source reference implementation, has a well documented format and API and has several other independent implementations. FLAC is freely available and supported on most operating systems

3.6 AAC format:

MPEG2 Advanced Audio Coding or MPEG4 Advanced Audio Coding is similar to MP3, although it's a bit more efficient. files that take up less space, AAC generally achieves better sound quality than MP3 at the same bit rate. It's almost as widely compatible with MP3. It is a proprietary audio coding standard for lossy digital audio compression.

AAC is a wideband audio coding algorithm that exploits two primary coding strategies to dramatically reduce the amount of data needed to represent high-quality digital audio:

- Signal components that are perceptually irrelevant are discarded.
- Redundancies in the coded audio signal are eliminated.

Chapter 4

Principal Component Analysis

Principal Component Analysis is basically a data dimension - reduction technique which reduces large set of variables into smaller one which still contains most of the information in

the large set. It is a mathematical procedure that a number of possibly correlated variables into a smaller set of uncorrelated variables called principal components. If we have a large data matrix M, such that $M = L_0 + N_0$, and we want to extract low rank component $L_0$and small perturbation matrix N_0, both having arbitrary magnitudes, Principal Component Analysis can be used. PCA seeks best rank - k estimate of L by solving

$$\text{minimize} \quad ||M - L|| \qquad \text{...(4.1)}$$
$$\text{subject to} \quad rank(L) \leq k$$

$||.||$ indicates 2-norm; that is, the largest singular value. This problem can be efficiently solved by Singular Value Decomposition (SVD)[3].

A Principal Component Analysis can be considered as a rotation of the axes of the original variable coordinate system to new orthogonal axes, called principal axes, such that the new axes coincide with directions of maximum variation of the original observation[9].

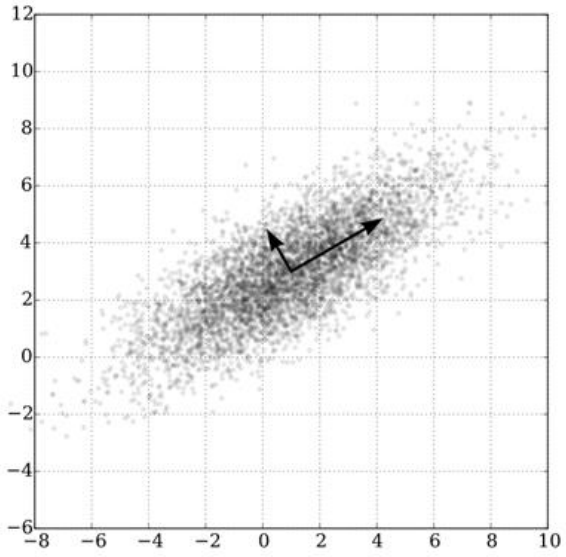


Figure 4.1: Direction of Principal Components

The property of maximum variation of the projected points defines the principal axis; it is the line or direction with maximum variation of the projected values of the original data points. The projected values corresponding to this direction of maximum variation are the principal components.

First Principal Component (PCA1):

The first principal component is the linear combination of variables that have maximum variance (among all linear combinations). It accounts for as much variation in the data as possible.

Second Principal Component (PCA2):

The second principal component is the linear combination of variables that accounts for as much of the remaining variation as possible, with the constraint that the correlation between the first and second component is 0

Interpretation of the Principal Components:

To interpret each component, compute the correlations between the original data and each principal component. Use the correlations between the principal components and the original variables to interpret these principal components. Because of standardization, all principal components will have mean 0. The standard deviation is the square root of the eigenvalue.

There is zero correlation between the components. Interpretation of the principal components is based on finding which variables are most strongly correlated with each component, i.e., which of these numbers are large in magnitude, the farthest from zero in either direction. Which numbers we consider to be large or small is of course is a subjective decision.

The disadvantage of PCA is that it is sensitive to outliers. One grossly corrupt entry in M could change value of estimates far from original values.

Chapter 5

Methodology

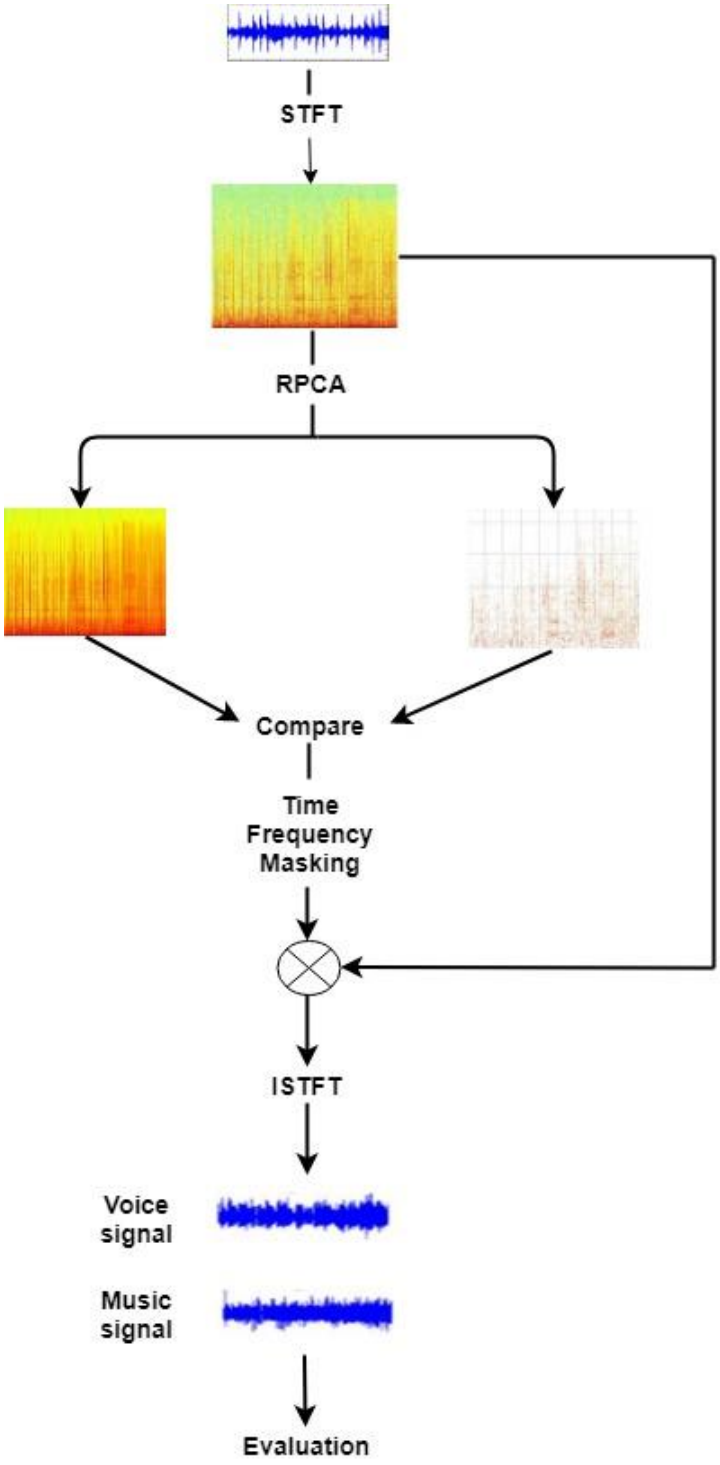


Fig. :

5.1 Short Time Fourier Transform (STFT):

Discrete Fourier Transform cannot provide simultaneous time and frequency localization.It provides excellent localization in the frequency domain but poor localization in the time domain. Hence it is not useful for analyzing time-variant non-stationary signal.

DFT provides the information about which frequencies exits but it does not provide any information about where these frequencies are located in time. But when we deal with audio signals we need time information as well. Hence Short Time Fourier Transform is preferred.
The basic idea of STFT is intuitive and simple. Slice the signal into different segments (with possible overlap) so as to gain time localization and subject each slice to fourier transform[10].

5.1.1 Windowing:

Slicing is equivalent to windowing a signal with finite-width window function w(t) since we need time localization. The equation can be written as,

$$x(\tau,t) = x(t) * w(t-\tau) \quad ...(5.1)$$

where τ is the center of the real-value, symmetric window function.
The regular fourier transform can be imagined as the short time fourier transform. In fact, it is a long time fourier transform, the window functions are infinite in length. One of the key requirements of the window function is that, it should be real valued and symmetric. The window function is placed in such a way that the center of this window will coincide with the start of the signal. So, that means, initially$\tau$is 0 . And then, entire signal length is traversed.
The window function should have compact support. This is a technical term to indicate, that the window exist only over a finite period of time and vanishes outside this interval.

$$x(\tau) = \begin{cases} x(t)w(t-\tau), & for\ t\ near\ \tau \\ 0\ , & for\ t\ far\ away\ from\ \tau \end{cases} \quad \ldots(5.1)$$

The rectangular window causes abrupt beginnings and ending of a segment. This abrupt beginning and end of a segment introduces spurious frequencies, that is, it causes spectral leakage. But, if a bell shaped or a smoothly tapered window function is used, there are no abrupt changes in the beginning and the end of the segment. Therefore, spectral leakage comes down. The major advantage of choosing this bell shape like window functions over the rectangular window, is that the spectral leakage is going to be mitigated.
The rectangular window has a very narrow bandwidth and very broad lobes. These side lobes should ideally decay to 0 very quickly for less spectral leakage. The longer these side lobes take to decay, the more the spectral leakage. In case of Hamming and Hanning, the side lobes roll of very quickly.
For a rectangular window with N observations, after performing DFT the resolution is $\frac{1}{N}$in frequency. But when Hamming or Hanning window is used, less importance is given to the samples in the beginning and in the end.

Narrower the window, better is its time resolving, time localizing ability of the energy, but then it loses out on the ability to resolve frequencies because the bandwidth is going to be much larger of the window. Wider windows will result in poor time localization, but gives better frequency localization.

Different windows are available for calculating short time fourier transform. Even though rectangular window provides best resolution but it is prone to spectral leakage. When using a bell shaped or a nice smoothly tapered window function such as hamming or Hanning window, then no abrupt changes in the beginning and the end of the segment are introduced. Therefore, the spectral leakage reduces.

5.1.1.1 Rectangular window

The (zero-centered) rectangular window may be defined by

$$□(□) = 1 \quad \text{...(5.3)}$$

where *M* is the window length in samples. As value of *M* increases, the main lobe narrows (better frequency resolution). It is also called as boxcar or Dirichlet window.

The rectangular window(fig. 3.2) is the simplest window, It is equivalent to replacing all but *N* values of a data sequence by zeros, making it appear as though the waveform suddenly turns on and off[11].

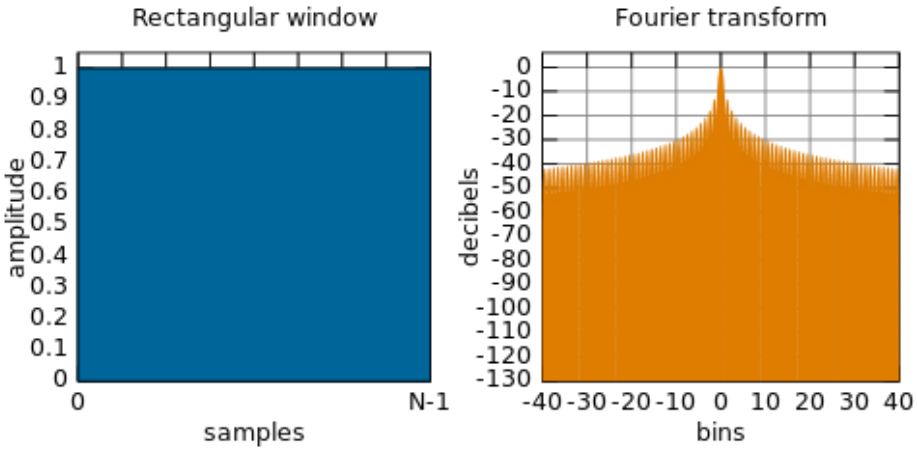


Figure 5.1: Rectangular window

It provides poor channel isolation ($\simeq 13$ dB). The rectangular window causes abrupt beginnings and ending of a segment. This abrupt beginning and end of a segment introduces spurious frequencies which introduces a lot of spectral leakage[12].

5.1.1.2 Hanning Window

It is also known as raised cosine, because the zero-phase version, is one lobe of an elevated cosine function. On the interval $n \in [0, M-1]$ the Hann window function is,

$$w(n) = 0.5[1 - \cos(\frac{2\pi n}{M-1})] \quad ...(5.4)$$

where M is the window length. The function is plotted in fig. 3.3.

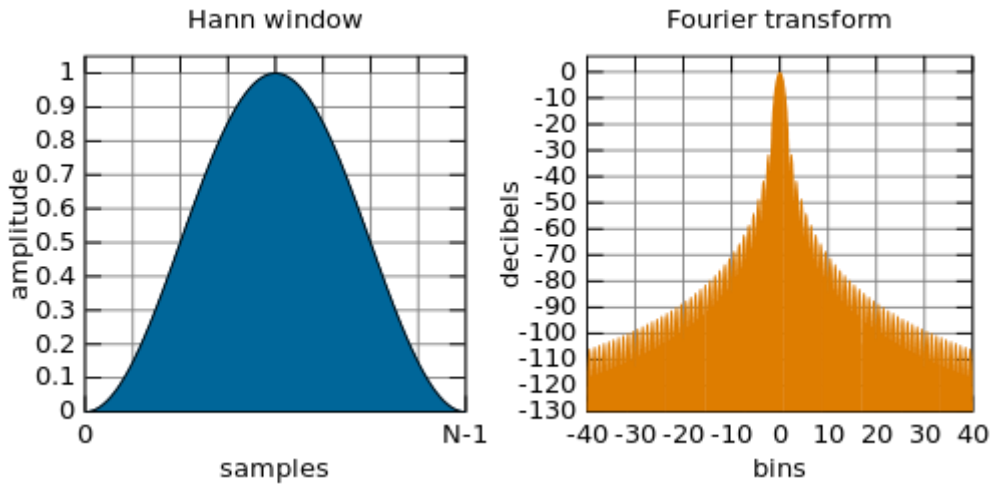


Figure 5.2: Hanning window[11]

The endpoints of the Hanning window just touch zero. Although the Hanning window does a good job of forcing the ends to zero, it also adds distortion to the waveform being analyzed in the form of amplitude modulation i.e., the variation in amplitude of the signal over the time record. In case of Hanning window, the sidebands, or side lobes effectively reduce the frequency resolution of the analyzer by 50%. The Hanning window should always be used with continuous signals, but must never be used with transients. This is because the window shape will distort the shape of the transient, and the frequency and phase content of a transient is intimately connected with its shape[13].

5.1.1.3 Hamming Window

The window is optimized to minimize the maximum (nearest) side lobe, giving it a height of about one-fifth that of the Hann window (fig. 3.4). Window function is given by,

$$w(n) = a - b * \cos(\frac{2\pi n}{M-1}) \quad ...(5.5)$$

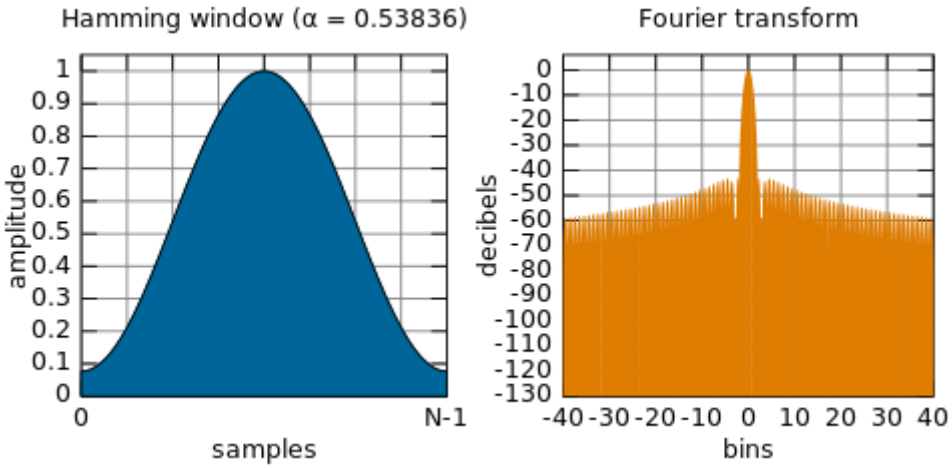


Figure 5.3: Hamming window[11]

Approximation of the constants to two decimal places substantially lowers the level of side lobes, to a nearly equi-ripple condition. In the equi-ripple sense, the optimal values for the coefficients are $\alpha = 0.53836$ and $\beta = 0.46164$. Its function is similar, but has its first side lobes 42 dB down, whereas the Hanning window's first side lobes are only 32 dB down. Thus the Hamming has better selectivity for large signals, but it suffers from the disadvantage that the rest of the side lobes are higher, and in fact fall off slowly at 20 dB per octave like those of the rectangular window[14].

5.1.2 Spectrogram:

A spectrogram is a visual way of representing the signal strength, or "loudness", of a signal over time at various frequencies present in a particular waveform. It gives the energy density of the signal in the time frequency plane. Spectrograms are basically two-dimensional graphs, with a third dimension represented by colors. Time runs from left (oldest) to right (youngest) along the horizontal axis. The vertical axis represents frequency, which can also be thought of as pitch or tone, with the lowest frequencies at the bottom and the highest frequencies at the top. The amplitude or energy of a particular frequency at a particular time is represented by the third dimension, color, with dark blues corresponding to low amplitudes and brighter colors up through red corresponding to progressively stronger (or louder) amplitudes.

Energy decomposition of the signal achieved by STFT in the T-F plane is,

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\Pi} \int_{-\infty}^{\infty} |X(\omega)|^2 d\omega \quad \ldots. (5.6)$$

The two dimensional area under the surface of the spectrogram gives the total energy[10].

5.2 Singular Value Decomposition:

Singular value decomposition takes a rectangular matrix A of n × p in which the n rows represent the cepstral coefficients, and the p columns represent the time sequence. SVD is represented as,

$$A_{nxp} = U_{nxn} \sum_{nxp} V^T_{pxp}. \quad ...(5.7)$$

where $□^□□ = □_{□×□}$, $□^□□ = □_{□×□}$; the columns of □ are orthonormal eigenvectors of $□□^□$, the columns of □ are orthonormal eigenvectors of $□^□□$, and ∑ is a diagonal matrix containing the square roots of eigenvalues from □ or □ in descending order.

5.2.1 Eigenvalues and Eigenvectors:

Eigenvalues of any square matrix A are the roots of characteristic equation of that matrix.

Let $A = [a_{jk}]$ be a given nonzero square matrix of dimension $n \times n$. Consider the following vector equation,

$$Ax = \lambda x \quad ...(5.8)$$

The problem of finding non-zero x's and λ's that satisfy this equation is called an eigenvalue problem. A is a given square matrix, x is an unknown vector, and λ is an unknown scalar. A value of λ for which this equation has solution $x \neq 0$, is called an eigenvalue or characteristic value of the matrix A. The corresponding solutions $x \neq 0$are called the eigenvectors or characteristics vectors of A corresponding to that eigenvalue λ.Eigenvalue decomposition finds similarity transformation to diagonal form[15].

5.2.2 Singular values:

Eigenvalues play an important role where the matrix is transformation from one vector space onto itself. Singular values play an important role where the matrix is a transformation from one vector space to a different vector space, possibly with a different dimension.

Singular values σ_{ii} are the positive square roots of eigenvalues of A^TA or AA^T. If there are n singular values, they are obtained such that $\sigma_{11} \geq \sigma_{22} \geq \ldots \geq \sigma_{nn}$.

5.2.3 Dimensionality reduction using SVD:

$$A = U\sum V^T = \sum_{i=1}^{n} \quad (u_i \sigma_{ii} v_i^T) \quad ...(5.9)$$

For dimensionality reduction, use only k dimensions, that is, retain first k columns for U and $V$and first k values of $\sum$. First k columns of Vgive the basis vectors in reduced space. Best rank k approximation in terms of sum squared error is given as,

$$A \approx \sum_{i=1}^{k} \quad (u_i \sigma_{ii} v_i^T) = U_{1\ldots k} \sum_{1\ldots k} V_{1\ldots k}^T \qquad \ldots(5.10)$$

$$T \approx AV_{1\ldots k} \qquad \ldots(5.11)$$

Total energy of reduced matrix is sum of squares of singular values (known as spread or variance),

$$E = \sum_{i=1}^{n} \quad \sigma_{ii}^2 \qquad \ldots(5.12)$$

If k dimensions are retained, p% energy is retained as shown below,

$$E_k = \sum_{i=1}^{k} \quad \sigma_{ii}^2 \qquad \ldots(5.13)$$

$$E_k / E \geq p$$

Generally, p is between 80% to 95% [16].

5.3 Augmented Lagrangian Multiplier (ALM):

The convex optimization problem of RPCA can be solved using Augmented Lagrangian Multiplier (ALM) method[4]. Exact ALM (EALM) and Inexact ALM (IALM) are the two types of ALM. Other methods such as Iterative Thresholding, Accelerated Proximal Gradient (APG) can also be used for separating sparse and low-rank matrix. ALM is found to be more accurate than other methods.

The exact ALM has good convergence speed as compared to APD. Inexact ALM is the slight improvement of exact ALM. It has same convergence speed as that of the exact ALM, but the required number of partial SVDs (Singular Value Decompositions) is significantly less. IALM is five time faster than APG, and its precision is also higher[4].

5.4 Masking:

Given the separation result of the low-rank □ and sparse □ matrices, we can further apply binary time-frequency masking methods for better separation results. We define binary time-frequency masking □□ as follows[2]

$$M_b(m,n) = \begin{cases} 1 & |S(m,n)| > gain * |L(m,n)| \\ 0 & otherwise \end{cases} \qquad \ldots (5.14)$$

for all □ = *1* ... □*1* and □ = *1* ... □*2*

Once the time-frequency mask M_b is computed, it is applied to original STFT matrix M to obtain the separation matrix $M_{singing}$ and M_{music}.

$$\begin{cases} X_{singing}(m,n) \\ X_{music}(m,n) \end{cases} = \begin{matrix} M_b(m,n)M(m,n) \\ (1-M_b(m,n))M(m,n) \end{matrix} \quad \text{...(5.15)}$$

for all $m = 1 \ldots m_1$ and $n = 1 \ldots m_2$[3].

5.5 Inverse Short Time Fourier Transform:

The original time domain sequence can be constructed by applying inverse STFT. The "overlap-add" reconstruction method is one of the most robust method for doing this. In the frequency domain many copies of degraded signals can be merged in magnitude and phase, and modifications can be done to remove uncorrelated noise. This improves the signal to noise ratio. This method is nonlinear, as the modifications depend on the signals themselves. In spite of this, reconstructed sound after applying inverse STFT is not distorted. Applications of inverse STFT include speech de-reverberation, speech processing along with data processing applications.

The inverse STFT can be calculated as follows:

$$x[n+m] = \frac{1}{w*w[n]}\left[\sum_{k=0}^{k-1} X[k,m] * e^{w*\frac{2n}{k}km}\right] \quad \text{...(5.16)}$$

where $w[n]$= analysis window

k = number of frequency samples

Chapter 6

Augmented Lagrangian Multiplier

6.1 Optimality Condition:

□ is a minimum of f if $0 \in \partial f(x)$, i.e., 0 is a subgradient of f

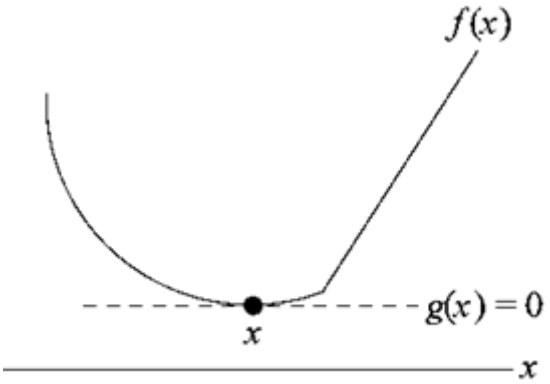


Figure 6.1: Plot of Optimality Condition

An ideal mathematical optimization problem is of the form

$$\text{minimize } f_0(\square) \qquad \text{...}$$

(6.1)

$$\text{Subject to } f_i(\square) \leq b_i, \quad i=1,....,m$$

Where the vector $\square=\{\square_1,....\square_n\}$ is the optimization variable of the problem

The function $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the objective function.

The function $f_i : \mathbb{R}^n \to \mathbb{R}$, i=1,...m are the inequality constraint function.

The constants b_1 ,...b_m are the limits or the bounds for the constraints.

In this problem, a vector □ is called optimal or solution of the problem if it has the smallest objective value among all other vectors that satisfy the constraints.

6.2 Unconstrained Optimisation:

Unconstrained optimization problems consider the problem of minimizing an objective function that depends on real variables with no restrictions on their values. Mathematically, let $x \in R_n$ be a real vector with n≥1 components and let $f: R_n \to R$ be a smooth function. Then, the unconstrained optimization problem is

$$min_x f(x) \quad ...(6.2)$$

Unconstrained optimization problems arise directly in some applications but they also arise indirectly from reformulations of constrained optimization problems. Often it is practical to replace the constraints of an optimization problem with penalized terms in the objective function and to solve the problem as an unconstrained problem.

Efficient algorithms for unconstrained global optimization use derivatives and partial derivatives to find local optima, to point out the direction in which moving from the current point does the most to increase or decrease the function. Such derivatives can sometimes be computed analytically, or they can be estimated numerically by taking the difference between values of nearby points. A variety of steepest descent and conjugate gradient methods to find local optima have been developed, similar in many ways to numerical root-finding algorithms[17][18].

6.3 Constrained Optimisation:

It is quite more difficult subject which makes use of more heavy duty linear algebra and geometry, and the results and algorithms are generally more complicated and technical.

Also finding points that satisfy all the constraints is often the difficult problem. One approach is to use a method for unconstrained optimization, but add a penalty according to how many constraints are violated. Determining the right penalty function is problem-specific, but it often makes sense to vary the penalties as optimization proceeds. At the end, the penalties should be very high to ensure that all constraints are satisfied.

After this procedure, attack the problem using any of the unconstrained methods that has been developed[17].

The constrained optimization problem is given by,

$$min_x \ f(x)$$
$$\text{subject to } c_j(x) = 0, j = 1, \ldots, m$$
...(6.3)

For a differentiable function f(x), its minimizer $\hat{x}$is given by

$$\frac{df(\hat{x})}{dx} = \left[\frac{\partial f(\hat{x})}{\partial x_1} \cdots \frac{\partial f(\hat{x})}{\partial x_m}\right]^T = 0$$
...(6.4)

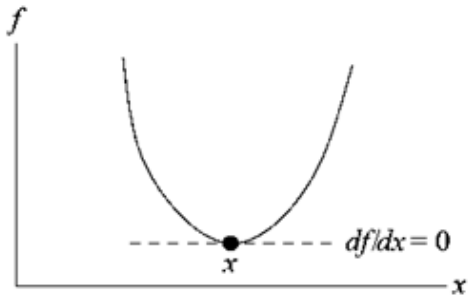


Figure 6.2: Plot of Differentiation of f(x)

6.4 Lagrange Method:

The solution of the optimization problem corresponds to a choice that has minimum cost (or maximum utility). A solution method for a class of optimization problems is an algorithm that computes a solution of the problem (to some given accuracy), given a particular problem from the class,i.e., an instance of the problem.

There is in general no analytical formula for the solution of convex optimization problems, but (as with linear programming problems) there are very effective methods for solving them.

The solution of a constrained optimization problem can often be found by using the so-called Lagrangian method. The Lagrangian can be defined as-

□(□,□) = □(□) + □ (□-□(□)) ...(6.5)

Lagrangian multipliers method formulates the convex optimization problem as

$$min_x \ f(x) + \sum_{j=1}^{m} \lambda_j c_j(x) \quad \text{...(6.6)}$$

The method of Lagrangian Multipliers was named after Joseph-Louis Lagrange. In mathematical optimization it is a strategy for finding the local maxima and minima of a function subject to equality constraints (constraint that the points on a certain surface are only considered).

In calculus one of the most common problems is that of finding maxima or minima of a function but it is difficult to find a closed form for the function being extremized. These kind of difficulties often arise when it is necessary to maximize or minimize a function subject to fixed outside equality constraints. The method of Lagrange multipliers is a powerful tool for solving this difficulties without the need to explicitly solve the conditions and use them to eliminate extra variables. The method of Lagrange multipliers can be extended to solve problems with multiple constraints using a similar argument. The Lagrangian method finds its applications in optimal control theory as well as nonlinear programming. In optimal control theory, the Lagrange multipliers are interpreted as costate variables, and Lagrange multipliers are reformulated as the minimization of the Hamiltonian, in Pontryagin's minimum principle. In nonlinear programming there are several multiplier rules, e.g., the Carathéodory-John multiplier Rule and the Convex Multiplier Rule for inequality constraints

6.5 Penalty Method:

In case of indirect method, the constrained problem must be transformed into an unconstrained problem. This is largely accomplished by adding to the objective function additional functions of the constraints that reflect very significantly the violation of the constraints. These functions are referred to as the penalty functions

Penalty method reformulates the problem as

$$min_k \; f(x) + \mu \sum_{j=1}^{m} c_j^2(x) \quad \text{.... (6.7)}$$

Here, the parameter μ increases over iteration and to get good solutions $\mu \to \infty$.

Penalty function methods approximate a constrained problem by an unconstrained problem structured such that minimization favors satisfaction of the constraints.

The general technique is to add to the objective function a term that produces a high cost for violation of constraints.

The advantage of this penalty function approach is that a 'hands-off' method for converting constrained problems of any type into unconstrained problems. Also many constraints in the real world need not be satisfied precisely. The penalty method is best to these kind of problem. But the drawback to penalty function method is that solution to the unconstrained penalized

problem will not be an exact solution to the original problem. In some cases penalty methods can't be applied because the objective function is actually undefined outside the feasible set. Another drawback to penalty methods is that if the penalty parameters are increased to more strictly enforce the constraints, the unconstrained formulation becomes very ill conditioned, with large gradients and abrupt function changes[19].

6.6 Augmented Lagrangian Multiplier method:

ALM is an indirect method for solving a constrained optimization problem. Any of the methods of the unconstrained optimization can be used to determine the minimum of the constrained problem. Indirect methods were produced so that advantages of codes for solving unconstrained optimization can be utilised. They are also referred to as Sequential Unconstrained Minimization Techniques (SUMT). The main idea behind this approach is to repeatedly call the unconstrained optimization algorithm using the results of the previous iteration to start the current one. The unconstrained algorithm itself executes many iterations

ALM is the most robust of the penalty function methods. More importantly it also provides information on the Lagrange multipliers at the solution. This is achieved by not solving for the multipliers but merely updating them during successive SUMT iterations. It overcomes many of the difficulties associated with the penalty function formulation without any significant overhead[17].

Augmented Lagrangian multipliers method combines Lagrange multipliers and penalty:

$$min_x \ f(x) + \sum_{j=1}^{m} \lambda_j c_j(x) + \frac{\mu}{2} \sum_{j=1}^{m} c_j^2(x) \quad \text{...(6.8)}$$

Here $\lambda = [\lambda_1 \dots \lambda_m]^T, c(x) = [c_1(x) \dots c_m(x)]^T$

The above problem can be reformulated as

$$min_x \ f(x) + \lambda^T c(x) + \frac{\mu}{2} ||c(x)||^2 \quad \text{... (6.9)}$$

After each iteration, alongwith updating μ_k variable $\lambda$is also updated based on the rule

$$\lambda_i \leftarrow \lambda_i - \mu_k c_i(x_k) \quad \text{... (6.10)}$$

Where $x_k$is the solution to the unconstrained problem at the kth step, i.e. $x_k = argmin\Phi_k(x)$. The variable λ is an estimate of the Lagrange multiplier and the accuracy of this estimate improves at every step. The major advantage of the method is that unlike the

penalty method, it is not necessary to take $\mu \to \infty$ in order to solve the original constrained problem. Instead, because of the presence of the Lagrange multiplier term,μ can stay much smaller, thus avoiding ill-conditioning. The method can be extended to handle inequality constraints.

Augmented Lagrangian methods are a certain class of algorithms for solving constrained optimization problems. They have similarities to penalty methods in that they replace a constrained optimization problem by a series of unconstrained problems and add a penalty term to the objective; the difference is that the augmented Lagrangian method adds yet another term, designed to mimic a Lagrange multiplier. The augmented Lagrangian is not the same as the method of Lagrange multipliers.

6.6.1 Inexact Augmented Lagrangian Multiplier

Algorithm[4]:

Input: Observation matrix $D \in \mathbb{R}^{m \times n}, \lambda$.

1: $Y_0 = D/J(D);\ E_0 = 0;\ \mu_0 > 0;\ \rho > 1;\ k = 0.$

2: **while** not converged **do**

3: // Lines 4-5 solve $A_{k+1} = arg$

4: $U, S, V = svd\ (D - E_k + \mu_k^{-1} Y_k);$

5: $A_{k+1} = US_{\mu_k^{-1}}[S]V^T.$

6: // Line 7 solves $E_{k+1} = arg$

7: $E_{k+1} = S_{\lambda\mu_k^{-1}}\ [D - A_{k+1} + \mu_k^{-1} Y_k].$

8: $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1});\ \mu_{k+1} = \rho\mu_k.$

9: $k = k + 1.$

10: **end while**

Output: (A_k, E_k).

Chapter 7

Software Support : MATLAB

MATLAB is an open source, cross-platform numerical computational package and a high-level, numerically oriented programming language. It can be used for signal processing, statistical analysis, image enhancement, fluid dynamics simulation, numerical optimization and modeling, simulation of explicit and implicit dynamical systems and (if the corresponding toolbox is installed) symbolic manipulations. The MATLAB platform is optimized for solving engineering and scientific problems. The matrix based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. A vast library of prebuilt toolboxes lets you get started right away with algorithms essential to your domain. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

7.1 MATLAB Functions:

Matlab consists of many inbuilt functions, that helps to execute the main code without having to write any extra code for that specific operation. In MATLAB, functions are defined in separate files. The name of the file and of the function should be the same. Functions operate on variables within their own workspace, which is also called the local workspace, separate from the workspace you access at the MATLAB command prompt which is called the base workspace.

Apart from these, user defined functions can be made as per the requirement. These functions can perform independently with their own workspace with the parameters provided by the main code or functions it is called in.

7.2 Audio processing toolbox:

Audio is normal and best handled by MATLAB, when stored as a vector of samples, with each individual value being a double - precision floating point number. A sampled sound can be completely specified by the sequence of these numbers and the sample rate. Any operation that MATLAB can perform on vector, can be performed on audio. The audio vector can be loaded and saved in the same way as any other MATLAB variable. The maximum representable range of values in 16-bit format is between -32768 and +32767, but when converted to double precision it is scaled to lie between -1 or +1.

Audio System Toolbox provides algorithms and tools for the design, simulation, and desktop prototyping of audio processing systems. It enables low-latency signal streaming from and to audio interfaces, interactive parameter tuning, and automatic generation of audio plugins for digital audio workstations. Audio System Toolbox includes libraries of audio processing algorithms (such as filtering, equalization, dynamic range control, and reverberation), sources (such as audio oscillators and wavetable synthesizers), and measurements (such as A- and C-weighting)[20].

Chapter 8

Evaluation

8.1 BSS Evaluation:

In Blind Source Separation, separate performance measures are computed for estimated signal $\widehat{s_j}$by comparing it to true source signal s_j.The evaluation involves two steps. First, estimated signal is decomposed as follows,

$$\widehat{s_j} = s_{target} + e_{interf} + e_{noise} + e_{artif} \quad \ldots(8.1)$$

where s_{target} is a version of $s_j$modified by an allowed distortion, and e_{interf}, e_{noise}, and e_{artif} are respectively the interferences, noise and artifacts error terms. These four terms should represent the part of $\widehat{s_j}$ perceived as coming from the wanted source s_j , from other unwanted sources $(s_{j'})_{j' \neq j}$, from sensor noises $(n_i)_{1 \leq i \leq m}$ and from other causes. In second step, energy ratios are computed to evaluate relative amount of each of these terms either on the whole signal duration or on local frames.

Orthogonal Projections are used to decompose the estimated signal. Let us denote $\Pi(y_1, \ldots, y_k)$ the orthogonal projector onto the subspace spanned by the vectors $y_1, \ldots, y_k$. The projector is $T \times T$matrix, where T is length of these vectors. Following three orthogonal projectors are considered,

$$P_{s_j} = \Pi\{s_j\}, \quad \ldots(8.2)$$

$$P_s = \Pi\left\{(s_{j'})_{1 \leq j' \leq n}\right\} \quad \ldots(8.3)$$

$$P_{s,n} = \left\{(s_{j'})_{1 \leq j' \leq n},\ (n_i)_{1 \leq i \leq m}\right\} \quad \ldots(8.4)$$

$\widehat{s_j}$ is decomposed as follows

$$s_{target} = P_{s_j}\,\widehat{s_j}, \quad \ldots(8.5)$$

$$e_{interf} = P_s\,\widehat{s_j} - P_{s_j}\,\widehat{s_j}, \quad \ldots(8.6)$$

$$e_{noise} = P_{s,n}\,\widehat{s_j} - P_s\,\widehat{s_j}, \quad \ldots(8.7)$$

$$e_{artif} = \widehat{s_j} - P_{s,n}\,\widehat{s_j}. \quad \ldots(8.8)$$

Numerical performance criteria is defined by computing energy ratios in decibels[21].

1. Source to Distortion ratio,

$$SDR = 10log_{10}\frac{||s_{target}||^2}{||e_{interf} + e_{noise} + e_{artif}||^2},$$

…(8.9)

2. Source to Interference ratio,

$$SIR = 10log_{10}\frac{||s_{target}||^2}{||e_{interf}||^2},$$

…(8.10)

3. Source to Noise ratio,

$$SNR = 10log_{10}\frac{||s_{target} + e_{interf}||^2}{||e_{noise}||^2},$$

…(8.11)

4. Source to Artifacts ratio,

$$SAR = 10log_{10}\frac{||s_{target} + e_{interf} + e_{noise}||^2}{||e_{artif}||^2}$$

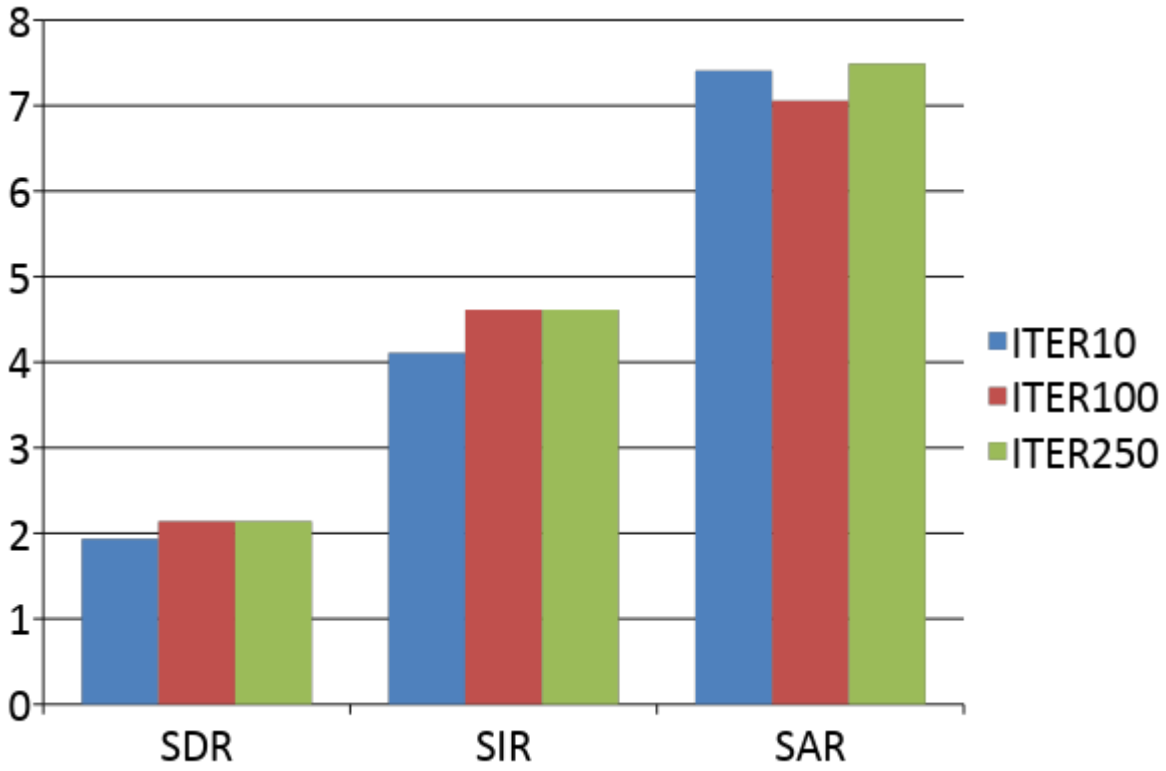
…(8.12)

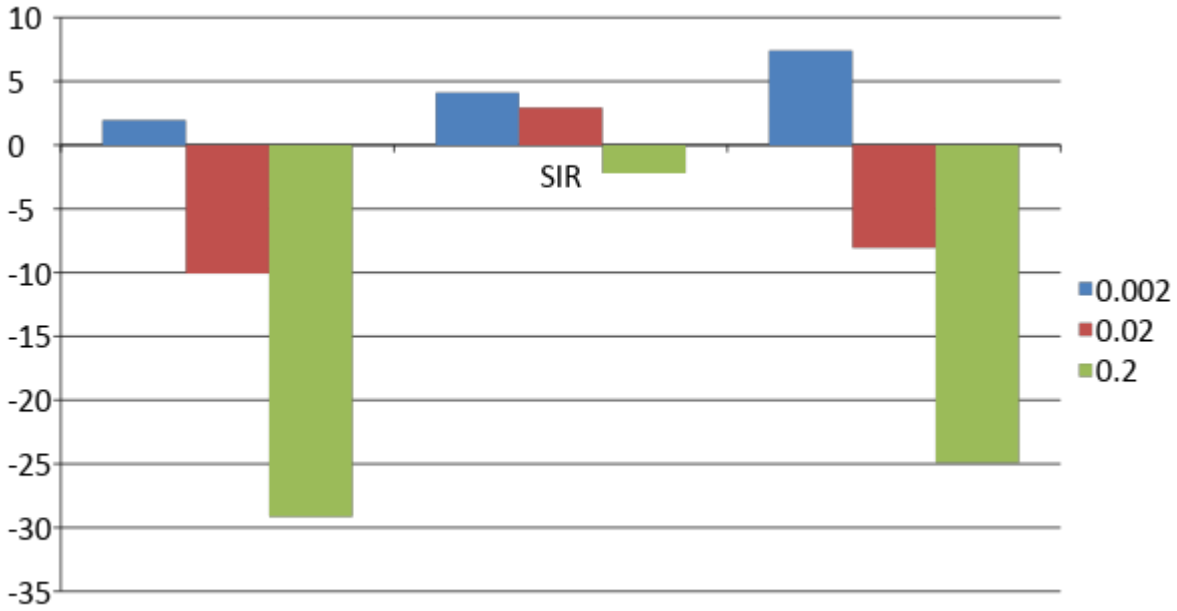
Chapter 9

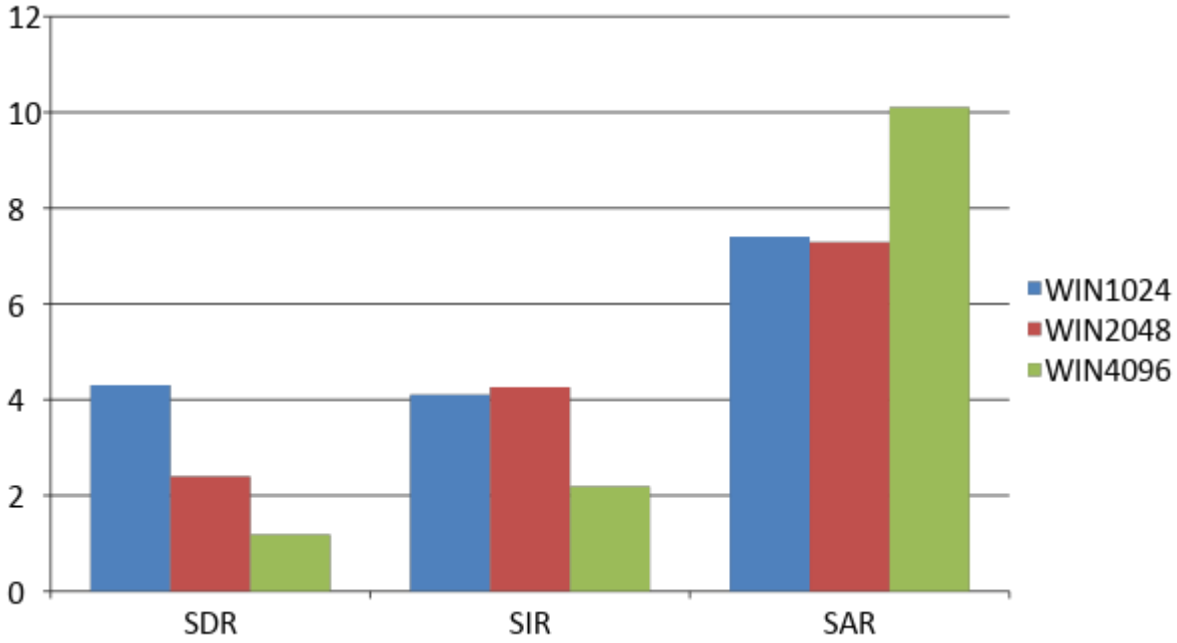
Results

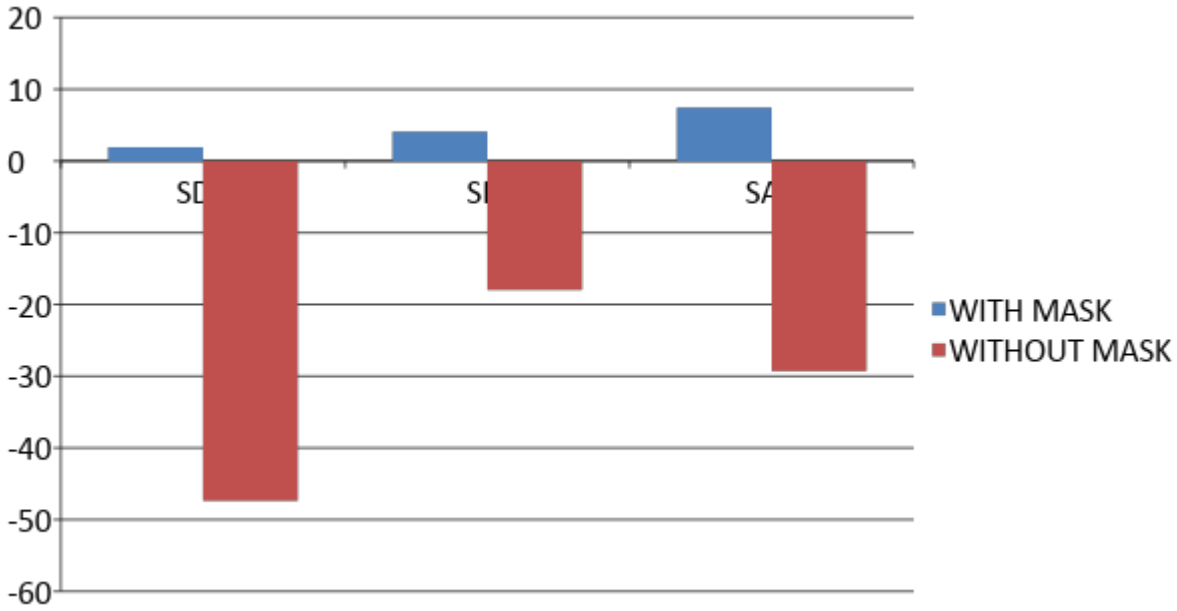
Chapter 10

Conclusion and Future Scope



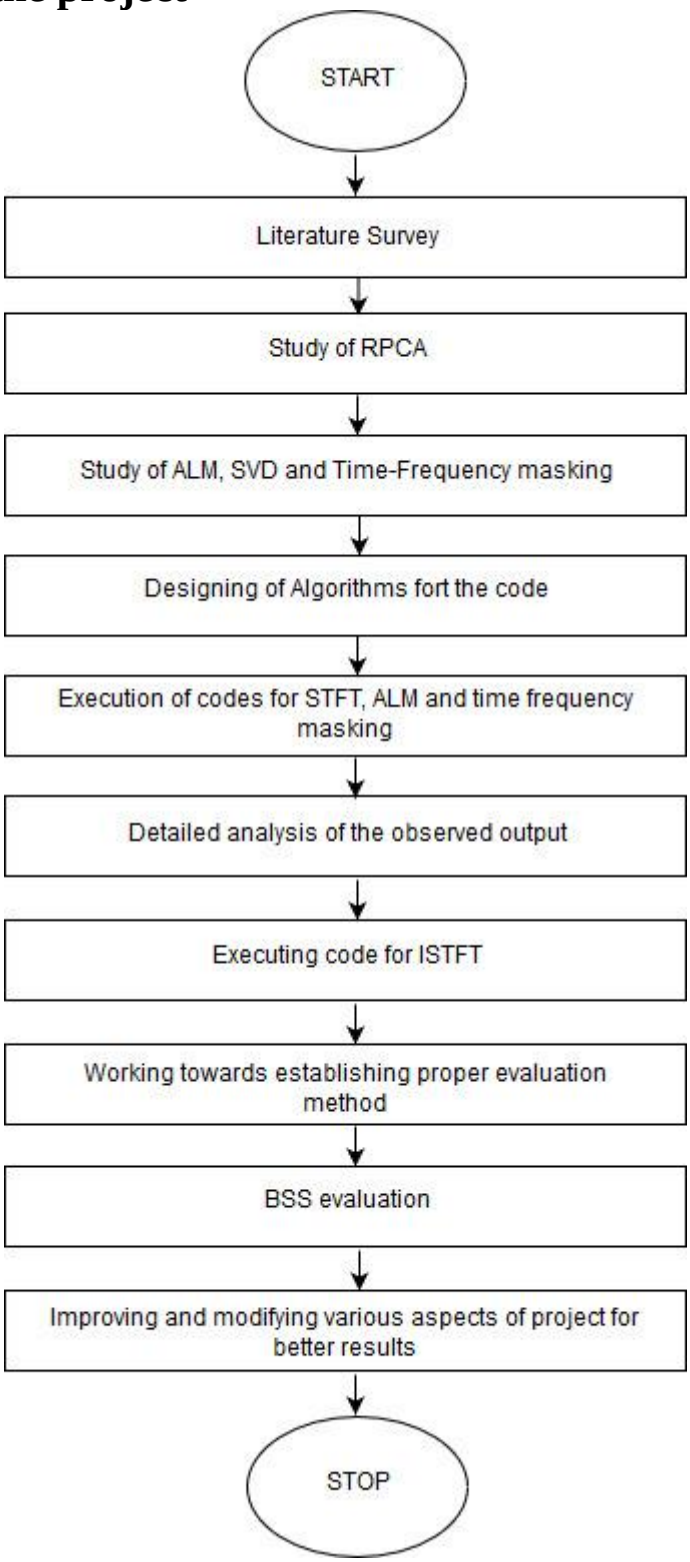

10
5
0
-5
-10
-15
-20
-25
-30
-35
SIR
0.002
0.02
0.2


12
10
8
6
4
2
0
SDR
SIR
SAR
WIN1024
WIN2048
WIN4096



Appendix A

Flowchart of the project



REFERENCES

[1] Björgvin Benediktsson, ‘All the EQ Information You’ll Ever Need in One Handy Chart’, 2017. [Online]. Available: http://www.citethisforme.com/guides/ieee-with-url/how-to-cite-a-website

[2] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, Mark Hasegawa-Johnson, “Singing-voice Separation From Monaural Recordings Using Robust Principal Component Analysis,” , IEEE conference on Audio, Speech and Language Processing, July 2012

[3] Emmanuel J. Candes, Xiaodong Li, Yi Ma, and John Wright, “Robust principal component analysis?,” J. ACM, vol. 58, pp. 11:1–11:37, Jun. 2011.

[4] Z. Lin, M. Chen, L. Wu, and Y. Ma, “The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices,” Tech. Rep. UILU-ENG-09-2215, UIUC, Nov. 2009.

[5] Zafar Rafii and Bryan Pardo. “REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 1, January 2013.

[6] Antoine Liutkus, Zafar Rafii, Roland Badeau, Bryan Pardo, and Gaël Richard. “Adaptive Filtering for Music/Voice Separation Exploiting the Repeating Musical Structure,” *37th IEEE International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan, March 25-30, 2012.

[7] Zafar Rafii and Bryan Pardo. “Online REPET-SIM for Real-time Speech Enhancement,” *38th IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, May 26-31, 2013.

[8] Zafar Rafii and Bryan Pardo. “Music/Voice Separation using the Similarity Matrix,” *13th International Society for Music Information Retrieval*, Porto, Portugal, October 8-12, 2012.

[9] ftp://statgen.ncsu.edu/pub/thorne/molevoclass/AtchleyOct19.pdf

[10] http://nptel.ac.in/courses/103106114/21

[11] Ricardo Gutierrez-Osuna, ‘Introduction to Speech Processing’, 2002. [Online]. Available: http://research.cs.tamu.edu/prism/lectures/sp/l6.pdf

[12] Stephane Boucher, ‘Spectral Audio Signal Processing’, 2015. [Online].Available: https://www.dsprelated.com/freebooks/sasp/STFT_Rectangular_Window_50.html

[13] Azima DLI, ‘Hanning Window’, 2009. [Online]. Available: http://azimadli.com/vibman/thehanningwindow.htm

[14] Azima DLI, ‘Hamming Window’, 2009. [Online]. Available: http://azimadli.com/vibman/gloss_hammingwindow1.htm

[15] Kreyszig E., Advanced Engineering Mathematics, 10th edition, John Wiley, 2011.

[16] http://nptel.ac.in/courses/106104021/pdf_lecture/lecture29.pdf

[17] https://people.rit.edu/pnveme/personal/EMEM820n/Mod6_Numerical/mod6_content/mod6_sec3_ALM.html

[18] https://neos-guide.org/content/unconstrained-optimization

[19] https://www.rose-hulman.edu/~bryan/lottamath/penalty.pdf

[20] https://in.mathworks.com/products/audio-system.html

[21] Cédric Févotte, Rémi Gribonval, Emmanuel Vincent. BSS_EVAL Toolbox User Guide – Revision 2.0. [Technical Report] 2005, pp.19. <inria-00564760>

Acknowledgement

The successful outcome of this project work titled: **'Extraction of Voice and Music from audio signal'** required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project.

We owe our deepest gratitude to our project guide Dr. Ravindra Chaudhari, Associate Professor of Electronics and Telecommunications Engineering Department for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

We would like to express our sincere thanks to our Director, Bro. Jose Thuruthiyil, **Head of Electronics and Telecommunication Dept., Dr. Gautam Shah**, and the entire Electronics and Telecommunication Dept. for their valuable comments and constructive criticism during various stages of this project.

In the end, our thanks and appreciation also go out to our parents and colleagues for providing us with moral support and encouragement.

Aishwarya Nambiar

Janak Pisharody

Swapnil Phalke

Sonali Parab