# Fundamentals of Audio Processing

Laurent Girin

PHELMA - SICOM 3A, M2 SIGMA

A large part of these slides is based on documents by Simon Leglaive - CentraleSupelec Rennes

# Introduction

What are audio signals?

What is audio processing?

What do we do in this course?

# What are audio signals?

- Basically: A recorded sound!
- Diversity of sounds / audio signals
  - Natural acoustic scenes (in diverse environments, indoor and outdoor)
  - Artificial or hybrid natural/artificial acoustic scenes (e.g., audio content of movies and video games)
  - Speech signals (a single one or a mixture of them)
  - Music signals (a single instrument or a mixture of them)
  - Industrial signals (sounds produced by machines)
  - Bioacoustic signals (e.g., whale songs)
  - Etc.

# What is audio processing?

- Audio signal analysis, modeling, synthesis and transformation
- Audio signal enhancement and restoration (noise reduction; noise estimation and compensation; bandwidth expansion)
- Audio source separation
- Audio source localization
- Acoustic scene analysis, sound events detection and classification. More generally: Automatic extraction of information from audio signals
- In the case of music signals, the latter is called Music Information Retrieval (MIR).
- Audio coding (compression)
- Psychoacoustics = analysis and modeling of Human audition (binaural hearing; auditory modeling; perceptual and psychophysical models; computational auditory scene analysis; signal processing for hearing aids and cochlear implants; etc.)

# A variety of methodologies

- Different methods for different tasks (e.g., enhancement in noise is quite different from classification)
- Single-channel vs multichannel processing (e.g., sound source separation)
  - Single-channel techniques are based on the spectral content of signals.
  - Multichannel techniques exploit the spatial diversity of multi-microphone recordings.
- Conventional methods based on signal and channel models vs modern methods based on deep learning

# About speech signals and music signals

## Speech is special

- A research/technological domain on its own (with the speech processing community and the audio processing community that are a bit too separated, IMO).
- Specific problems/tasks/applications: Automatic speech recognition (ASR), text-to-speech synthesis (TTS), speech language understanding, speaker diarization, etc.
- Strong link with natural language processing (NLP) and artificial intelligence (AI)
- A course dedicated to speech processing in SICOM 3A (Thomas Hueber)

## Music is special too

- Also a research/technological domain on its own (with a music processing community).
- Specific problems/tasks/applications: e.g., automatic transcription (signal-to-score conversion), musical sound synthesis (for a large set of instruments)
- Strong link with symbolic music processing and AI

# Examples of applications

- Speech enhancement for telecommunication applications (e.g., hand-free communication where the microphone is relatively far from the speaker's mouth)
- Audio quality improvement, e.g., restoration of old recordings
- Music production/editing
- Audio-based automatic surveillance (equivalent of video surveillance for audio)
- Artificial hearing for mobile/humanoid robots (companion robots)
- Diagnostic of machine breakdown
- Etc.

# Complementary ressources

- Tutorials, MOOCs, code, Githubs, etc. on the web.
- Books and scientific/technical papers (many references are provided along the slides)
- Top-level scientific journals in speech/audio processing
  - IEEE Transactions on Audio Speech and Language Processing
  - Journal of the Acoustical Society of America
- Top-level scientific conferences with proceedings
  - IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)
  - IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)
  - IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)

It is interesting to have a look at the subtopics of these different journals and conferences to have an idea of the variety of problems and techniques in acoustics, speech and audio processing.

# Objective, organization, and content of the course

Global objective: To give you the basic tools to understand and analyze audio signals.

Organization: 12h of course + 2 x 2h of "conference" + 4h of lab work

Content:

- Part 1: Basics of sound(s) and audio recordings
- Part 2: Fundamentals of audio analysis
- Part 3: Speech/audio denoising and separation
- Part 4: Spatial audio (with a focus on multichannel source separation)
- Conference 1: Deep generative models for musical sound synthesis, by Fanny Roche (Arturia)
- Conference 2: Fundamentals of Music Information Retrieval, by G. Peeters (Telecom ParisTech)

## Objective, organization, and content of the course

- Remark 1: It is impossible to investigate in details all aspects of audio processing in 20h, so the content of the course is somehow arbitrary. We will focus on a series of fundamentals that can be used in many different audio procesing problems.
- Remark 2: The course will focus on signal processing and machine learning aspects. Very few things on the physics of sounds.

# Part 1: Basics of sound(s) and audio recordings

# What is sound?

1. A characteristic of an "object"

   The sound of a guitar

2. An acoustic wave with physical properties that we can measure

   A sound of fundamental frequency 100 Hz

3. An auditory sensation

   A rich and bright sound

We will mostly focus on the second definition.

# What is sound (in the air)?

- A sound is generated by a vibrating object.
- Vibrations cause oscillations of air molecules, resulting in local regions of compression and rarefaction, i.e. modification of the acoustic pressure.
- The alternating pressure travels through the air as a longitudinal wave, from the source to the listener or to a microphone.
- In the case of a microphone, the acoustic wave is converted into an electrical signal.

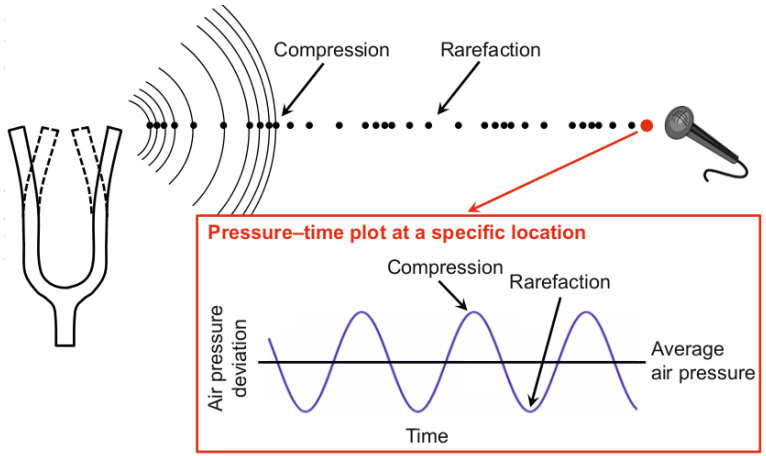



**Fig. 1.17** Vibrating tuning fork resulting in a back and forth vibration of the surrounding air particles. The pressure oscillation propagates as a longitudinal wave through the air. The waveform shows the deviation over time of the air pressure from the average air pressure at a specific location (as indicated by the microphone).

R. E. Berg & D. G. Stork, The physics of sound. Prentice Hall, 1995.

Image credit: M. Müller, Fundamentals of Music Processing, Springer, 2015.

# The waveform

- The change in air pressure at a certain location can be represented by a time-pressure plot, also referred to as the waveform of the sound.
- The simplest waveform: A pure tone is a simple sinusoidal wave. It is completely specified by its frequency $\nu$, amplitude $A$, and phase at origin $\phi$: $x(t) = A\cos(2\pi\nu t + \phi)$

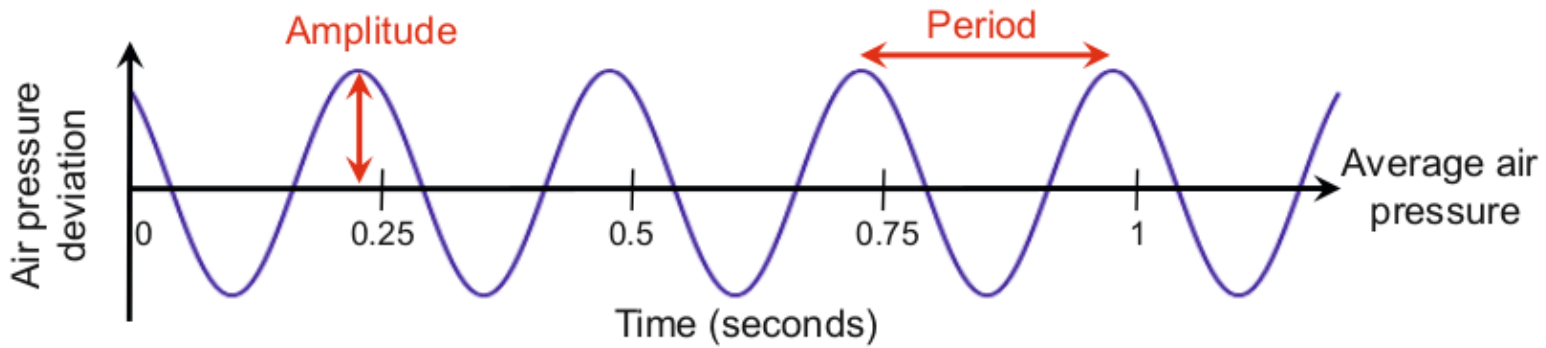



**Fig. 1.19** Waveform of a sinusoid with a frequency of 4 Hz.

Image credit: M. Müller, Fundamentals of Music Processing, Springer, 2015.

# Real world sounds... are generally much more complex than pure tones.

- They are composed of several spectral components (several "frequencies")
- Amplitude and frequency parameters are evolving with time (non-stationarity; modulations)

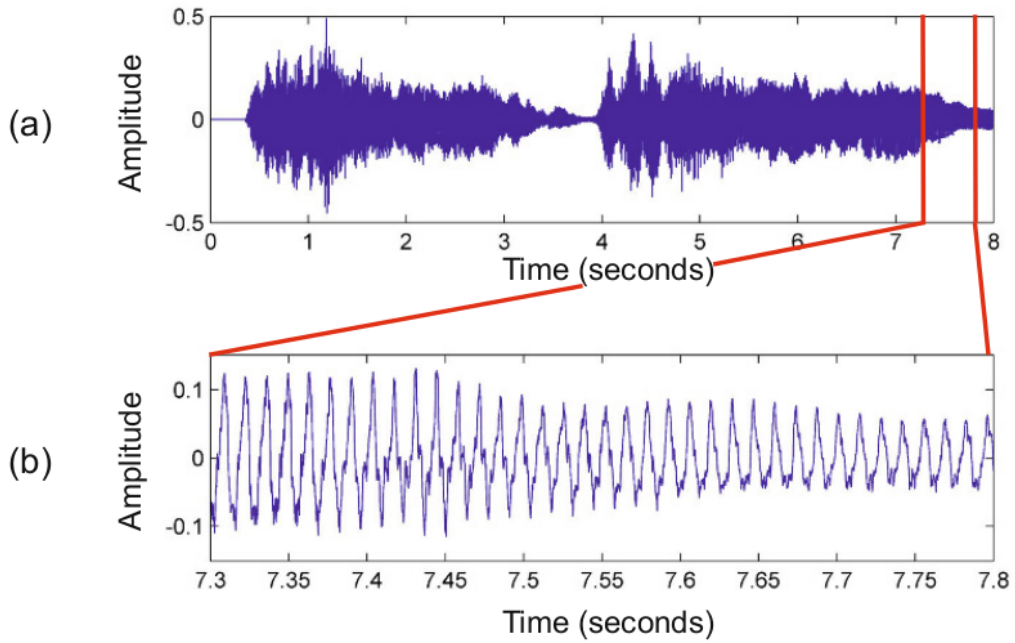
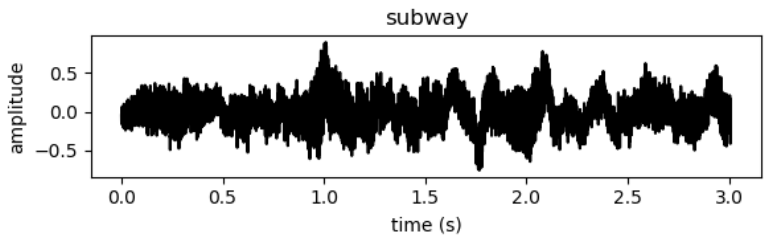
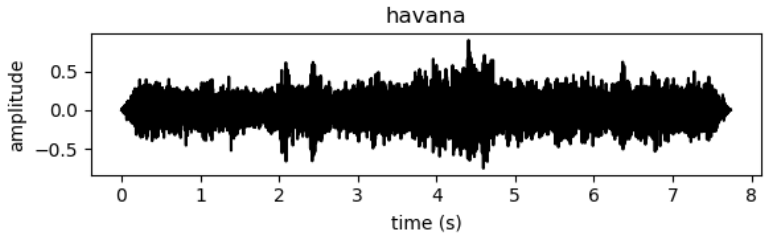


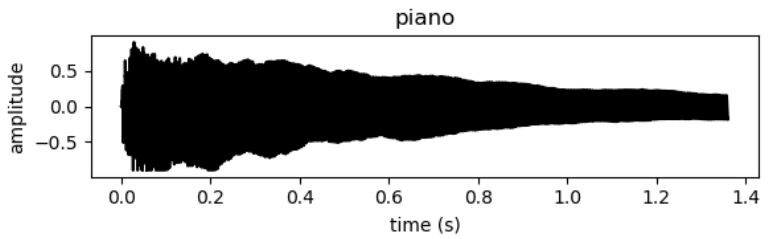

**Fig. 1.18** **(a)** Waveform of the first eight seconds of a recording of the first five measures of Beethoven's Fifth as indicated by Figure 1.1. **(b)** Enlargement of the section between 7.3 and 7.8 seconds.

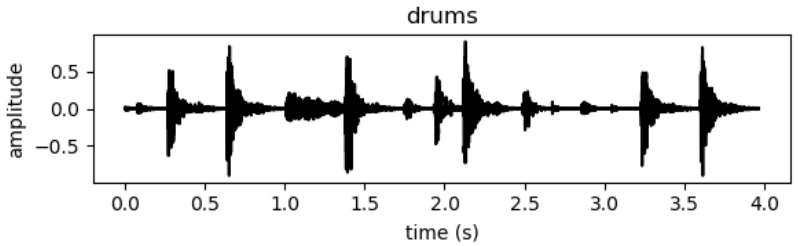
Image credit: M. Müller, Fundamentals of Music Processing, Springer, 2015.

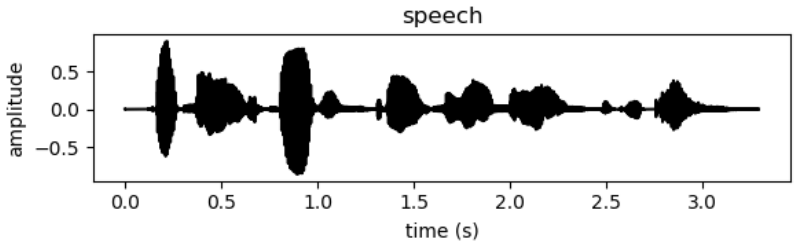
# Real world sounds... are generally much more complex than pure tones.

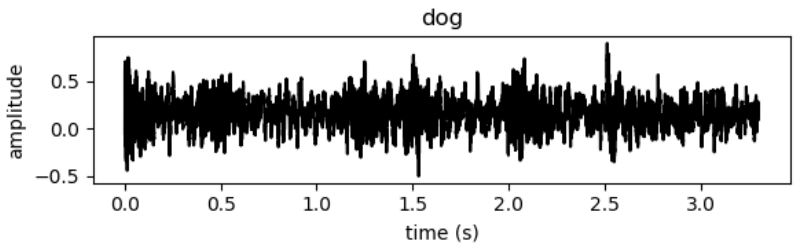



We need audio signal representation tools to analyze complex sounds.

# Bandwidth

- The perceived audio band (by Humans): #20Hz-20kHz
- This is an “approximate maximal” range, the actual range depends on individuals (age, health condition, personal history)
- Bandwidth of (physical/analog) speech signals: #70-10000Hz (and generally very few information above 8kHz)
- The bandwidth of (physical/analog) music signals is significantly larger, but in general “useful” natural sounds do not go well beyond the perceived audio band, because:
  - The physics of sound production and propagation are such that there is a rapid loss of energy as a function of frequency;
  - The evolution has shaped the Human ear (and the ear of other animals) in function of the environment.

# Dynamics

- Reminder: A sound results from a (rapid) variation of the acoustic pressure, which is expressed in Pa (Pascal) (1 Pa = 1 N/m²)
- Dynamical range = power ratio between the strongest and the weakest perceived sound
- Strict definition is narrowband (at a given frequency, because the Human ear perception threshold depends on frequency, see later)
- Can be extended to wideband signals (for a type of sound, see some examples in the next slide)
- Major example: The Sound Pressure Level (SPL), standardized by ANSI
  - Arbitrary standardized reference for the weakest signal: $P_{ref} = 20\mu\mathrm{Pa}$ at $\nu = 2\mathrm{kHz}$
  - $\mathrm{SPL} = 10\log_{10}\frac{P^2}{P_{ref}^2} = 20\log_{10}\frac{P}{P_{ref}}$ (in dB)
- Log scale is used because the physical range of natural sounds is massive: up to #$10^7$ on a linear scale = $140\mathrm{dB}$ ! (each time you double the amplitude, you increase the SPL by $6\mathrm{dB}$.)

# Dynamical range of "natural" (wideband) sounds

| Sound source | $P$ (Pa RMS) | SPL (dB) |
|---|---|---|
| Theoretical limit (1 atm) | 100000 | 194 |
| Pain threshold | 100 | 134 |
| Danger (short-term exposure) | 20 | 120 |
| Long haul plane (at 100m) | 6–200 | 110–140 |
| Jackhammer (at 1m)<br>Night-club loudspeaker (at 1m) | 2 | 100 |
| Dense car traffic<br>Danger (long-term exposure) | 0.2–0.6 | 80-90 |
| Moving car (at 10m)<br>TV (at 1m) | 0.02–0.2 | 60-80 |
| Normal conversation (at 1m) | 0.002–0.02 | 40–60 |
| Very quiet room | 0.0002–0.0006 | 20–30 |
| Calm breathing | 0.00006 | 10 |
| Perception threshold at 2kHz | 0.00002 | 0 |

# Digital signal

- The recorded electrical signal is turned into a digital signal using an analog-to-digital converter (ADC), which applies sampling and quantization.

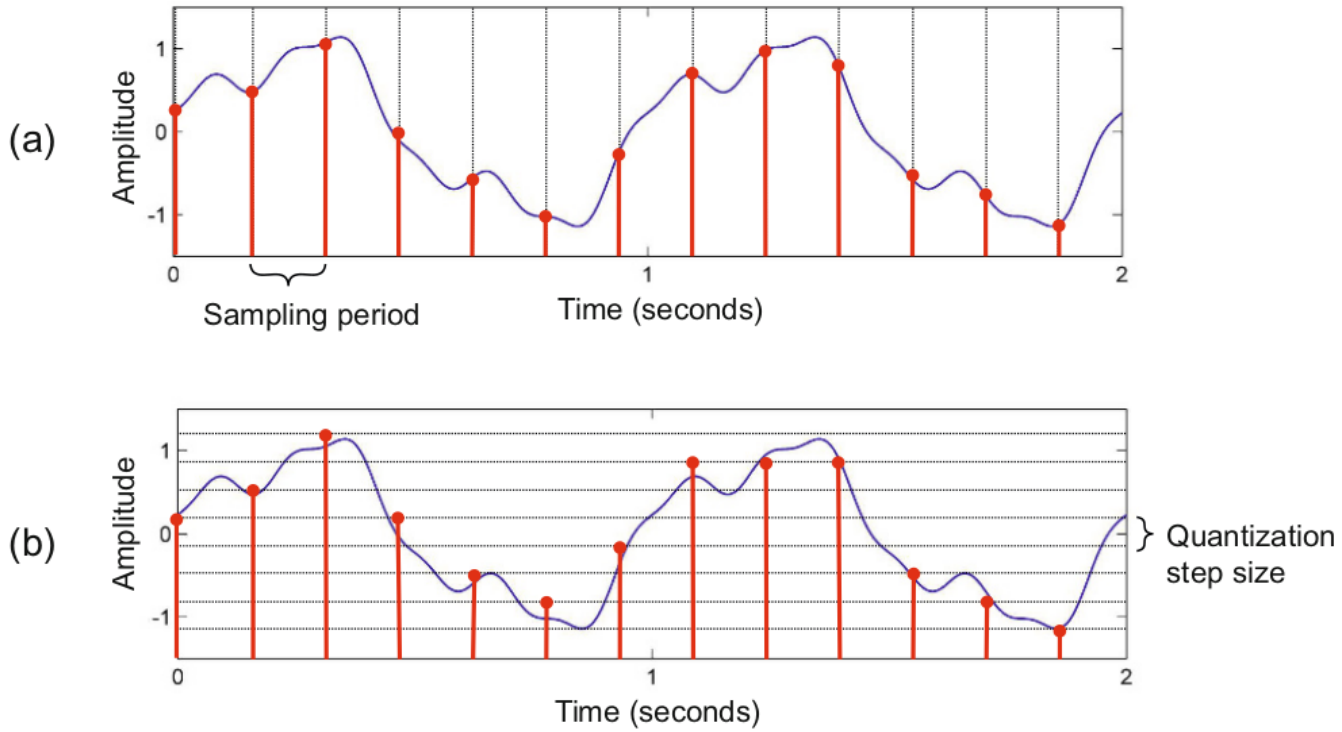



**Fig. 2.13** Two steps of a digitization process to transform an analog signal (solid curve) into a digital signal (stem plot). **(a)** Sampling. **(b)** Quantization.

Image credit: M. Müller, Fundamentals of Music Processing, Springer, 2015.

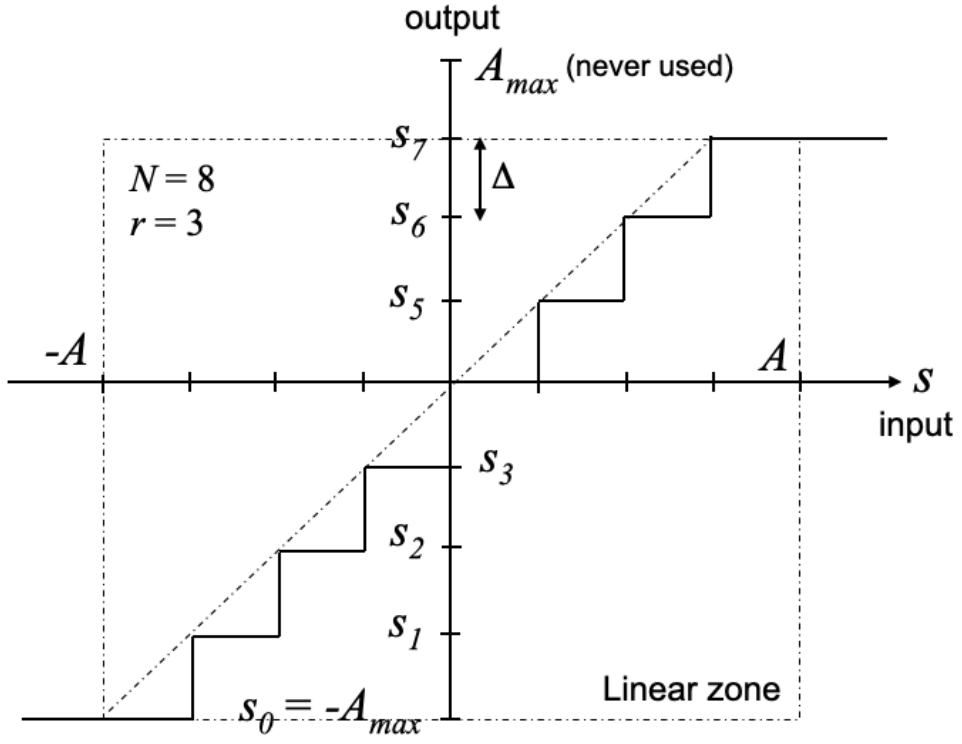
# Sampling of audio signals

- Sampling is the process of taking signal values every $T_s$ second to form a sequence of discrete values representing the continuous signal.
- The sampling rate $F_s = 1/T_s$ in Hz specifies the number of samples per second.
- Sampling theorem: To allow for perfect reconstruction of the analog signal, the sampling frequency must satisfy $F_s \geq 2\nu_{\max}$ where $\nu_{\max}$ is the highest frequency of the signal.
- $F_s/2$ is called the Nyquist frequency.
- Usual sampling frequency values in speech/audio processing:
  - $8\text{kHz}$ for speech signals in narrow-band telephony (bandwidth = $300$-$3400\text{Hz}$; intelligible but low-quality speech signals)
  - $16\text{kHz}$ for speech signals in wideband telephony (much better quality)
  - $44.1\text{kHz}$ for the compact-disk (CD)
  - $48$, $96$ or $192\text{kHz}$ for professional audio (!)

# Quantization of audio signals

- Quantization is the process of replacing signal amplitude continuous values with discrete (quantized) values from a predefined grid.
- Linear vs non-linear quantization (uniform or non-uniform distribution of the quantized values on the grid)
- The resolution $r$ is the number of bits used to encode the quantized values with a fixed-length binary code. It specifies the number of quantized values $N_r = 2^r$. For example, 16-bit coding for the CD implies $2^{16} = 65536$ possible quantized amplitude values.
- In linear quantization, the quantization step $\Delta$ is the size of a quantization cell = the maximal value of the quantization error.
- Let $A_{max}$ and $A_{min}$ be the maximal and minimal amplitude values on the quantization grid. Audio waveforms are centered, so that we generally have $A_{min} = -A_{max}$.
- We have: $\Delta = \frac{2A_{max}}{N_r} = \frac{A_{max}}{2^{r-1}}$

# Uniform scalar quantization

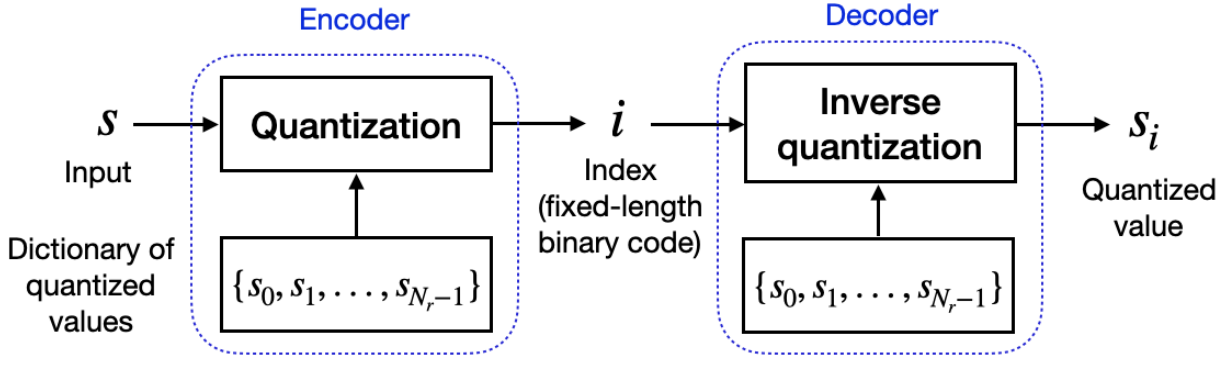



Note: This is a pedagogical schema. In practice, $N_r$ is large and $\Delta$ is small.

- The physical range $[-A, A]$ is distinct of the digital range $[-A_{max}, A_{max}]$. There are arbitrary gains in the ADC chain and the value of $A_{max}$ is arbitrary.
- Quantization is generally made towards the closer lower quantized value.
- Because $N_r$ is even, there is a slight dissymetry ($A_{max}$ is not part of the grid), but this has no practical consequence.
- If the signal gets out of the physical range $[-A, A]$, it is quantized at the min/max values = saturation (clipping) :-(

# Audio signals "formats"

- Do not confuse the quantized value and the code: A quantized value is obtained from the code by the decoding operation, aka inverse quantization.




- An audio file of a signal quantized with USQ contains the binary codes of the quantized audio samples, not the quantized values themselves.

- When reading such an audio file, the binary codes are decoded into signed or unsigned, integer or fixed-point values, depending on the "read" option. For example, from a $16$-bit audio file, we can decode signed integer values in $[-32768, 32767]$ (hence $\Delta = 1$). Dividing those values by the max $32768$ yields normalized values in $[-1, 1[$ ($\Delta = 1/32768$).

# Audio signals "formats"

- Single-channel vs 2-channel stereo (for most musical mixes) vs multichannel formats (e.g., home video, live music / studio recording, machine/robot audition)
- Uncompressed vs compressed formats: Audio coding for transmission and storage (if time permits, we will see an example of audio coding technology later)
- A widely used single/multichannel uncompressed format: the WAV format
  - Contains essential meta-information in a header, e.g., the sampling frequency $F_s$, followed by the signal sample codes
  - Readable/writable by all Digital Audio Workstations (DAW) and audio processing softwares, and most general processing softwares (e.g., Matlab)
- Remark: Some audio files, including WAV files, are in float format...

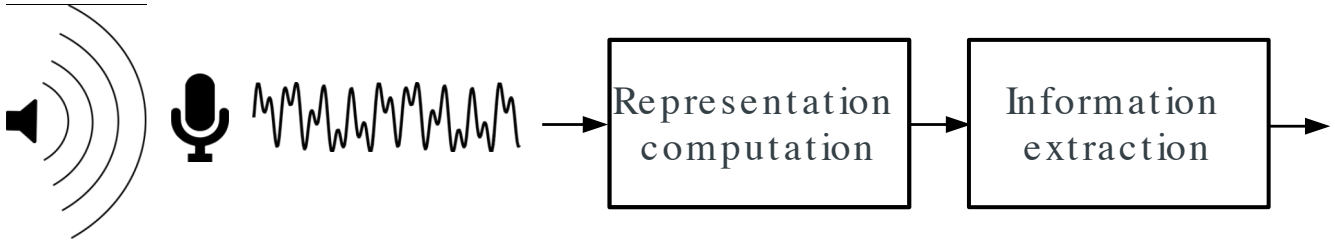
# Part 2: Fundamentals of audio analysis

# Content of Part 2

- Spectrogram representation of audio signals (generalities)
- The Discrete(-time) Fourier transform
- The Short-time Fourier transform

# Why do we want to analyze sounds?

- Transformation of existing sounds and creation of new ones (sound synthesis)
- Machine listening: Automatic retrieval of information from audio signals, and exploitation of this information to make a decision (i.e., solve a classification problem)
- This is often a two-step process:

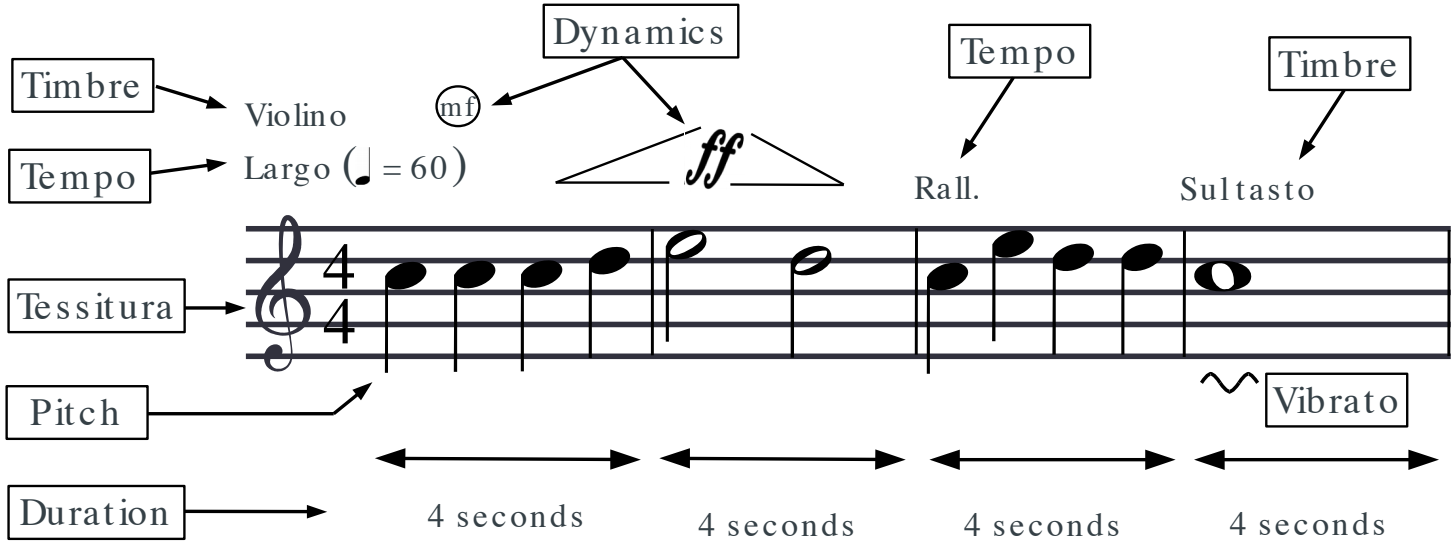



- voice activity detection
- automatic speech recognition
- automatic music transcription
- speech emotion recognition
- sound event classif cation
- etc.

- The goal of the representation (or feature) computation step is to ease the extraction of information with further signal processing / machine learning algorithm.
- We sometimes talk about low-level (signal) processing vs high(er)-level (information) processing.

# Towards a “meaningful” representation

- What are meaningful properties of audio signals?
- Let us look at what musicians use to represent the sounds of instruments: the musical score.
- A succession of “audio events” (notes) with indicators of pitch (related to melody), dynamics (related to intensity), tempo (related to time), and timbre (related to the distribution of energy in function of frequency - but not only).



# Introducing the (STFT) spectrogram

- Given the waveform of an audio signal, we would like to compute a representation highlighting the characteristics of the signal along these four dimensions.
- Such a representation is given by the spectrogram.
- Basically, a spectrogram is a representation of the distribution of signal energy as a function of frequency and time = a representation of the evolution of the signal spectrum as a function of time
- It is part of the general family of (or more or less equivalent to) time-frequency (TF) representation.
- We will see later in details how to compute a specific type of spectrogram that is EXTENSIVELY used in speech/audio processing: The Short-Term Fourier Transform (STFT).

# Automatic music transcription

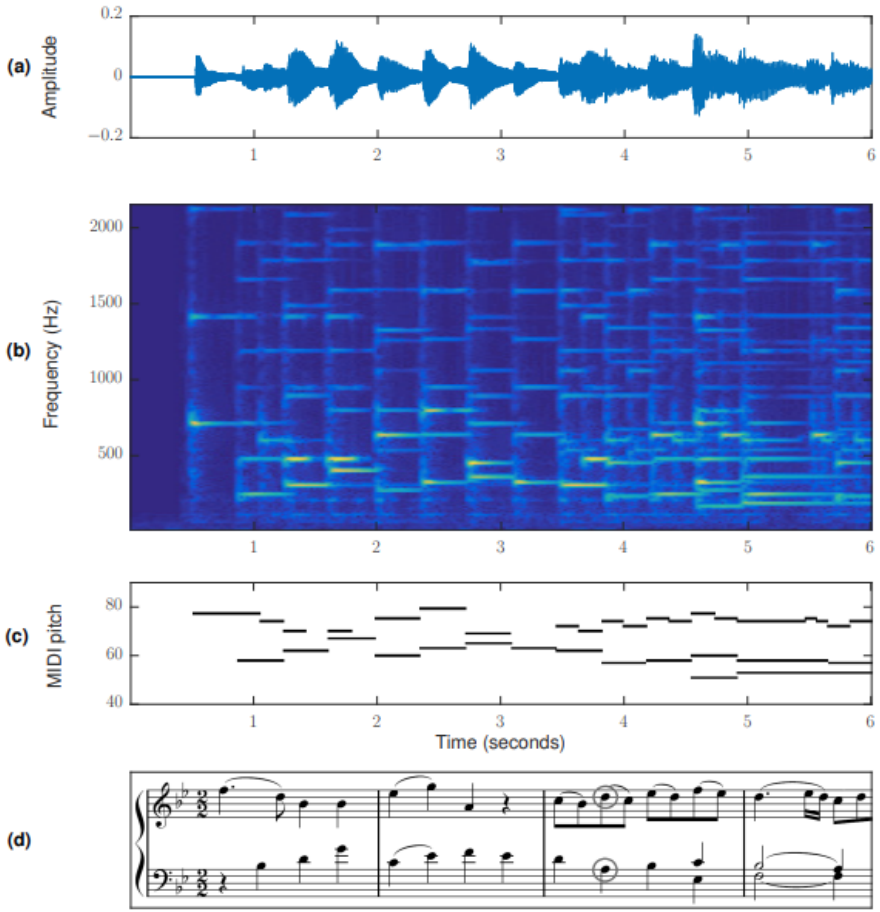



Figure 1. Data represented in an AMT system. (a) Input waveform, (b) Internal time-frequency representation, (c) Output piano-roll representation, (d) Output music score, with notes A and D marked in gray circles. The example corresponds to the first 6 seconds of W. A. Mozart's Piano Sonata No. 13, 3rd movement (taken from the MAPS database).

Image credits: E. Benedetos et al., Automatic Music Transcription: An Overview, IEEE Signal Processing Magazine, 2019.

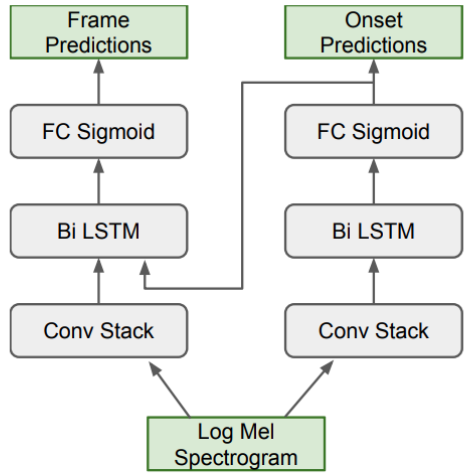



Figure 6. Google Brain's Onset and Frames Network: The input is processed by a first network detecting note onsets. The result is used as side information for a second network focused on estimating note lengths (adapted from [24]). *Bi LSTM* refers to bi-directional LSTM layers; *FC Sigmoid* refers to a fully connected sigmoid layer; *Conv Stack* refers to a series of convolutional layers.

C. Hawthorne et al., Onsets and frames: Dual-objective piano transcription, Conference of the International Society for Music Information Retrieval, 2018.

# Spectrograms are everywhere

# Automatic speech recognition

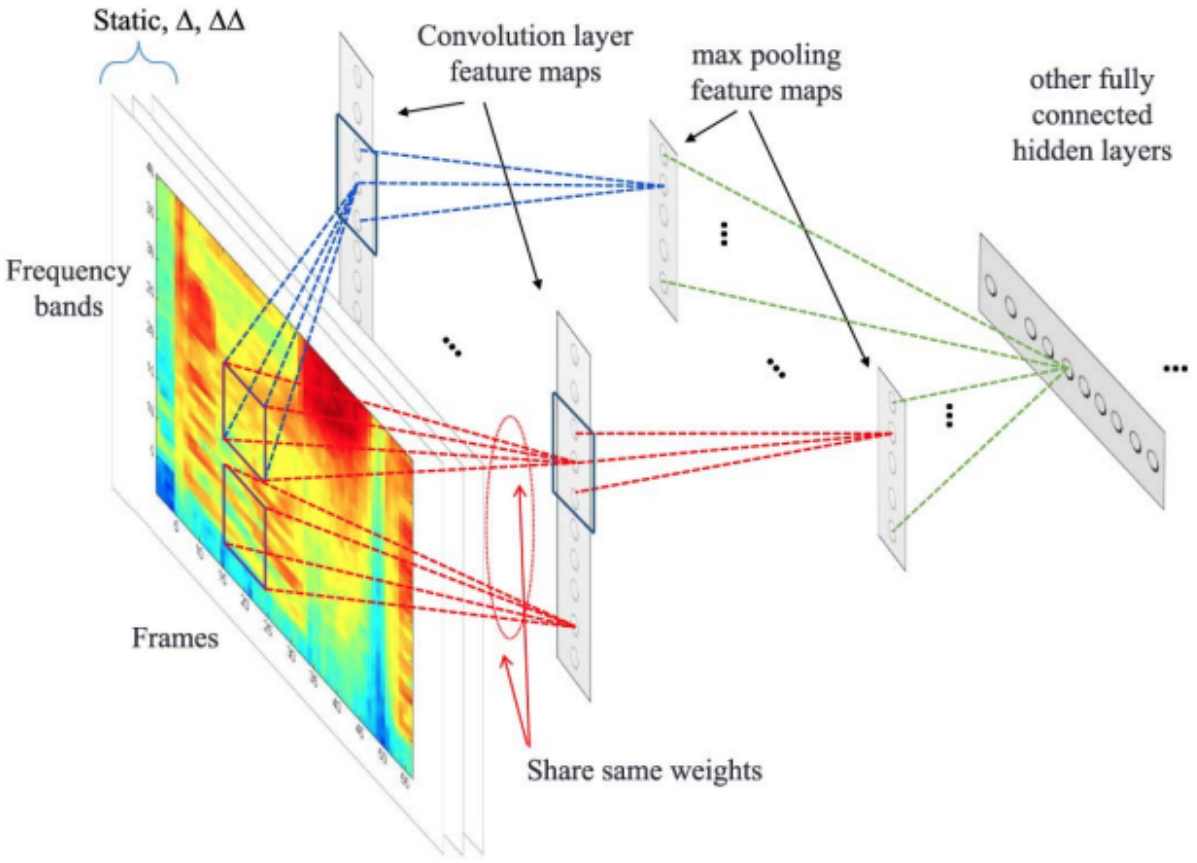
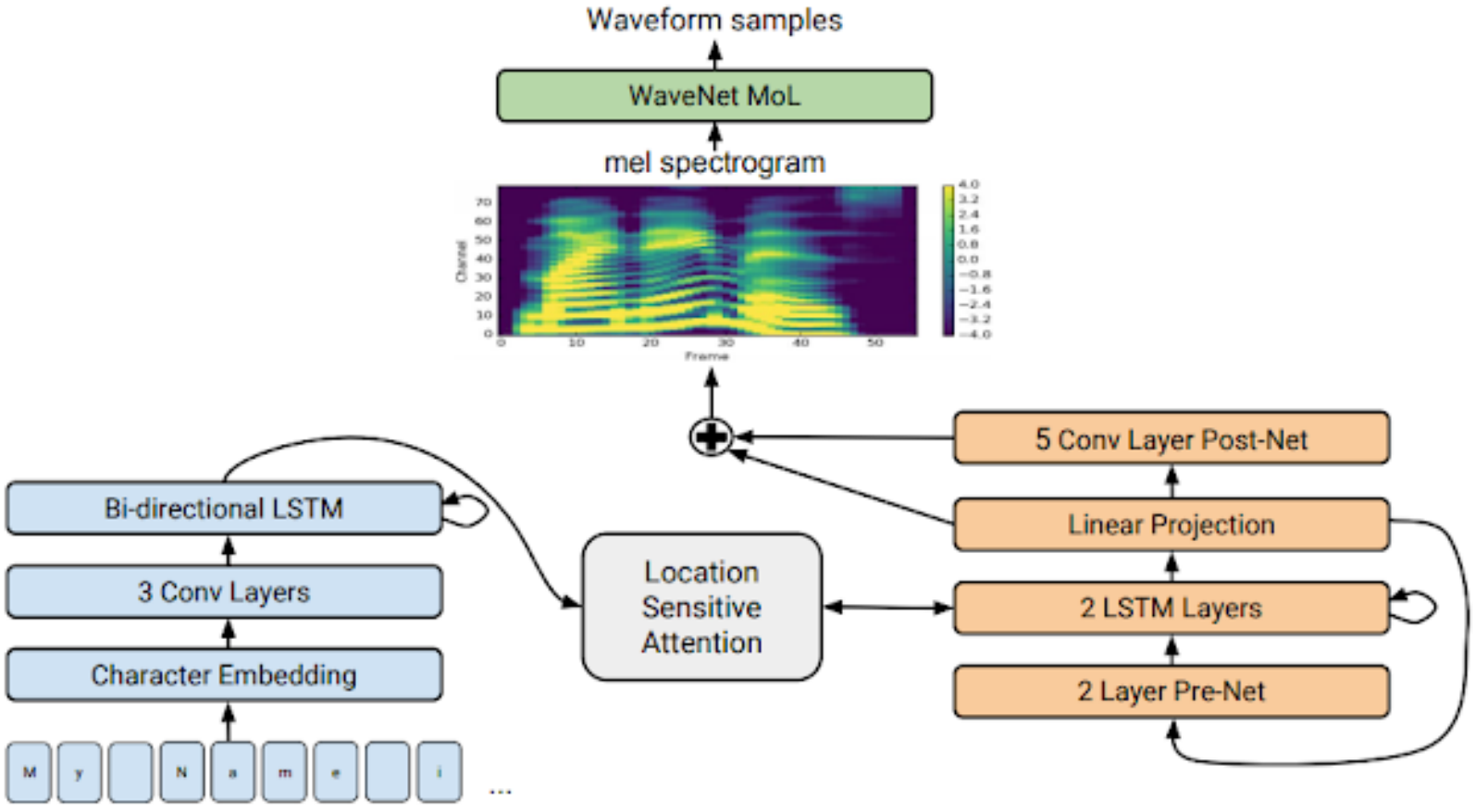



Fig. 3. An illustration of the regular CNN that uses so-called full weight sharing. Here, a 1-D convolution is applied along frequency bands.

Image credits: O. Abdel-Hamid et al., Convolutional neural networks for speech recognition, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2014.

# Text-to-speech synthesis

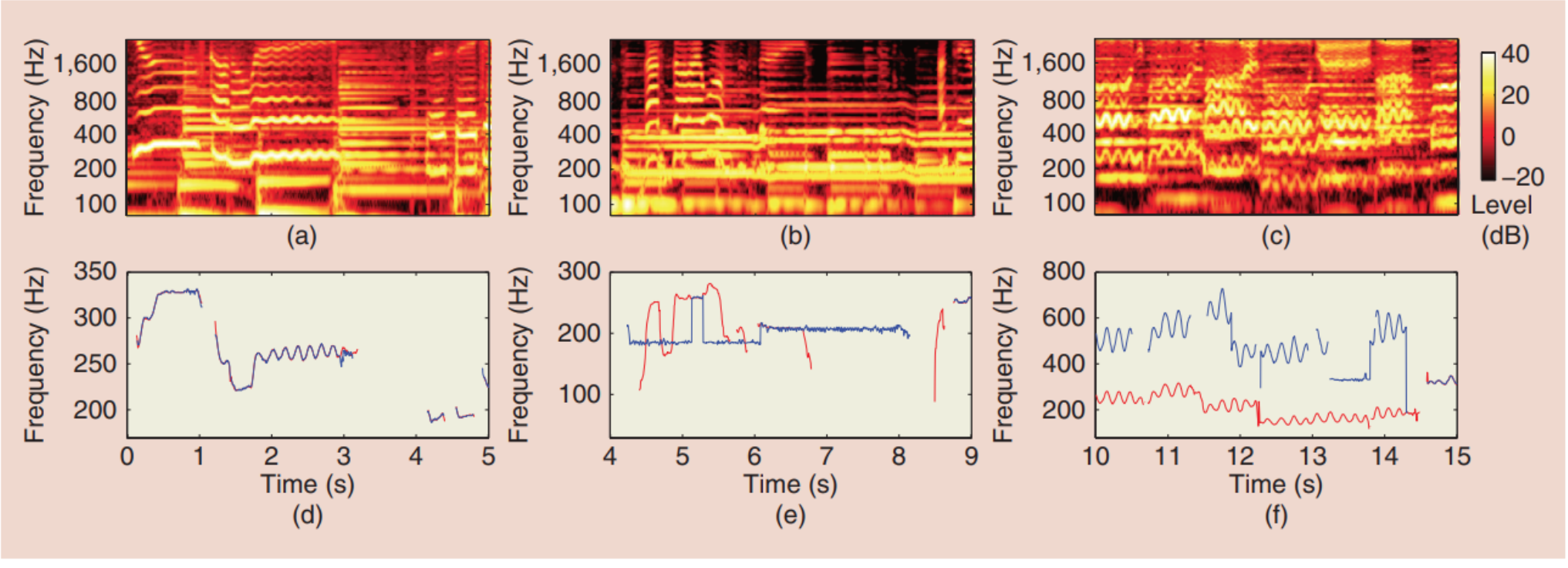



*A detailed look at Tacotron 2's model architecture. The lower half of the image describes the sequence-to-sequence model that maps a sequence of letters to a spectrogram. For technical details, please refer to the paper.*



Image credits: J. Shen and R. Pang, Tacotron 2: Generating Human-like Speech from Text, Google AI Blog, 2017.

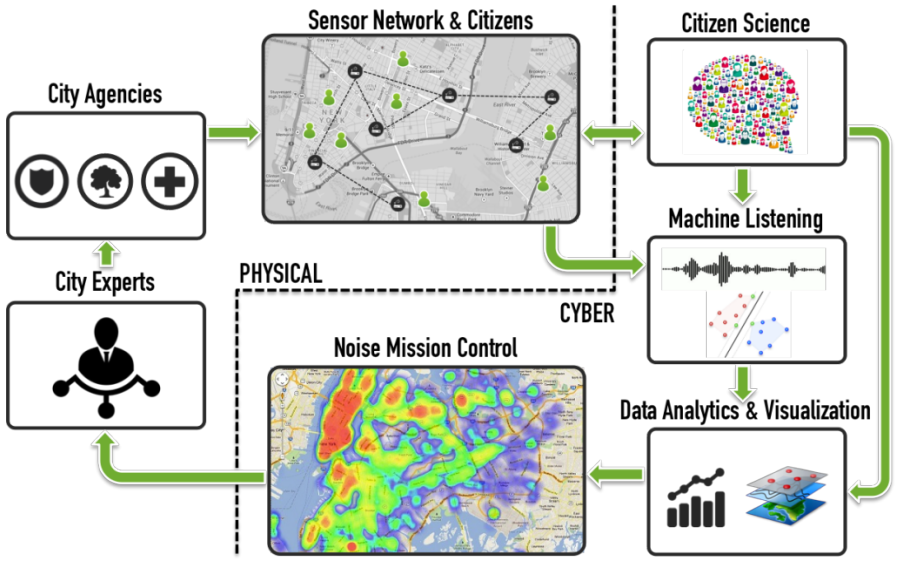
# Melody extraction in music




**[FIG2] Case study examples: (a)–(c) show the log-frequency spectrogram of three excerpts in the genres of (a) vocal jazz, (b) pop, and (c) opera. Parts (d)–(f) show the extracted melody [16] (blue) and ground truth (red) for each excerpt, respectively.**

Image credits: J. Salamon et al., Melody extraction from polyphonic music signals: Approaches, applications, and challenges, IEEE Signal Processing Magazine, 2014.

# Urban sound analysis



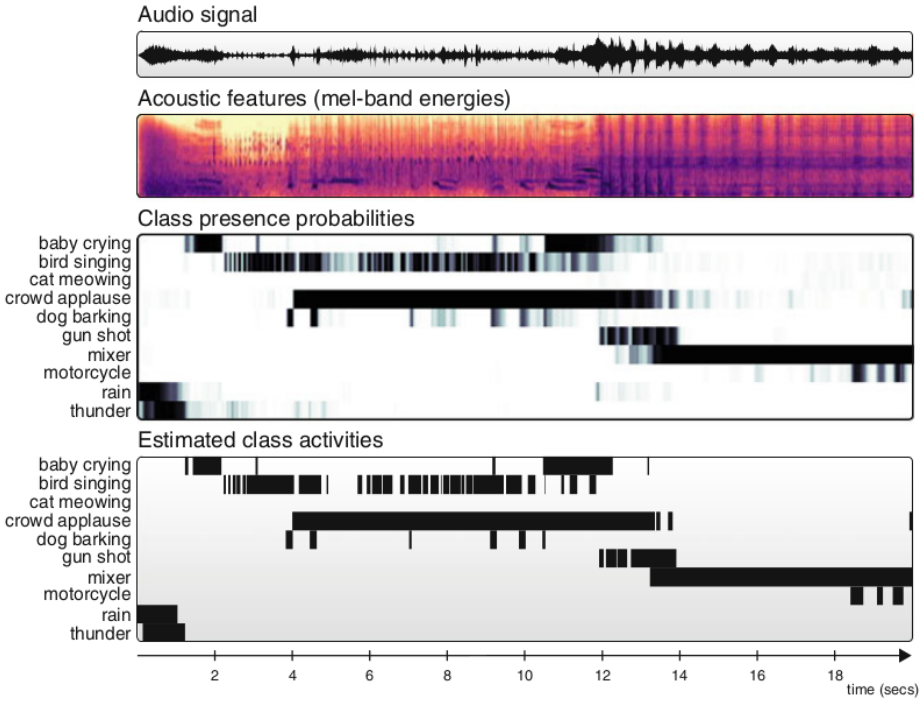



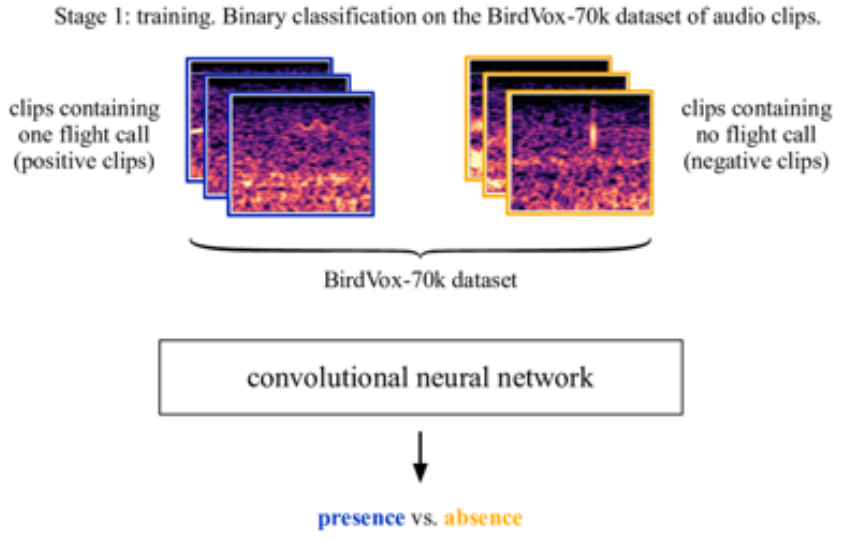

**Fig. 2.12** Illustration of acoustic features, class presence probabilities, and estimated class activities for a multi-label sound event detection task

Image credits:

Left: Sounds of New York City

Right: T. Heittola et al., The machine learning approach for analysis of sound scenes and events, in Computational Analysis of Sound Scenes and Events, Springer, 2018.

# Bird calls detection



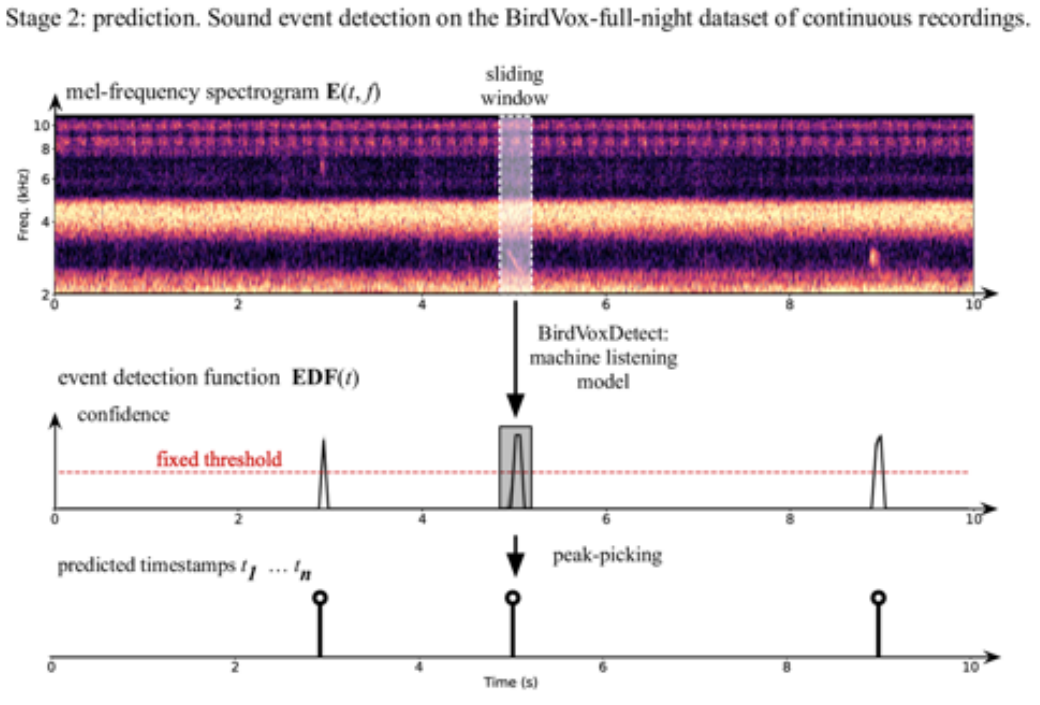
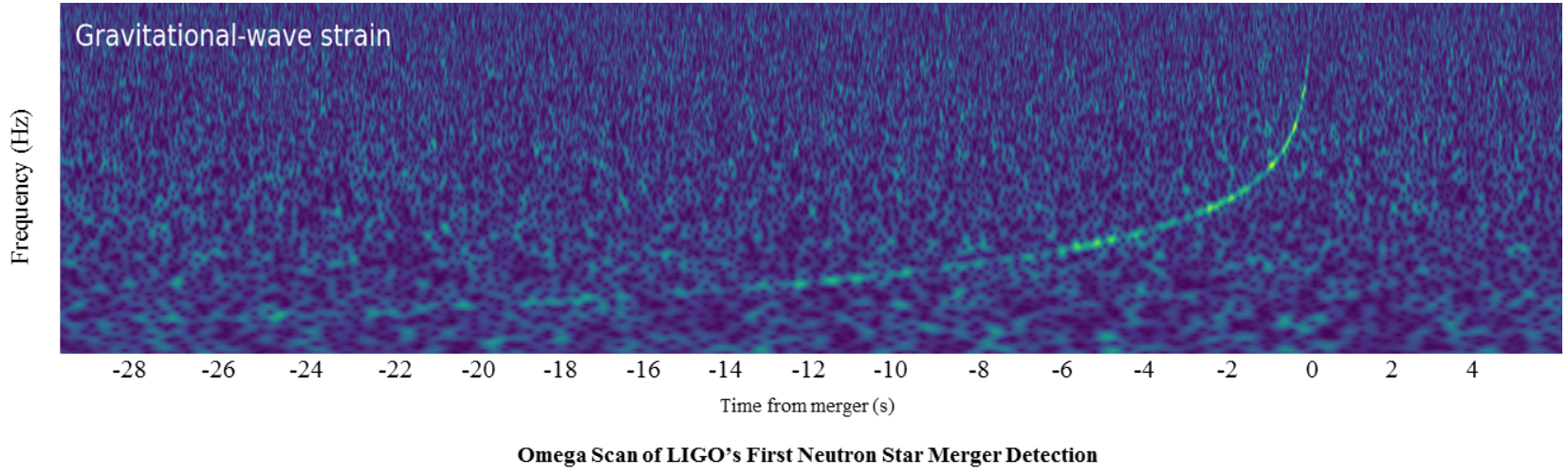




Image credits: V. Lostanlen et al., Robust Sound Event Detection in Bioacoustic Sensor Networks, PLoS ONE, 2019.

# Gravitational astronomy

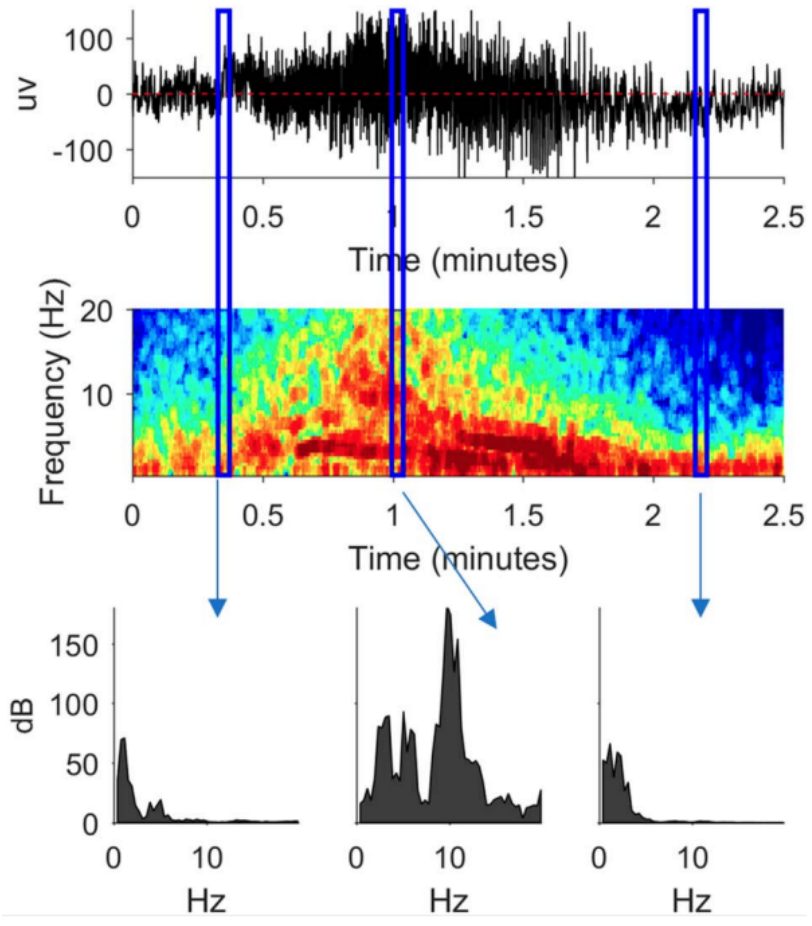



*Gravitational Wave Chirp Spectrogram*

This spectrogram combined the signals from both Hanford and Livingston detectors to show the characteristic sweeping chirp. As the neutron stars came closer to each other, circling faster, they produced higher frequency gravitational waves shown by the greenish line sweeping upwards.

Image credits: Caltech/MIT/LIGO Lab, https://www.ligo.caltech.edu/news/ligo20171016

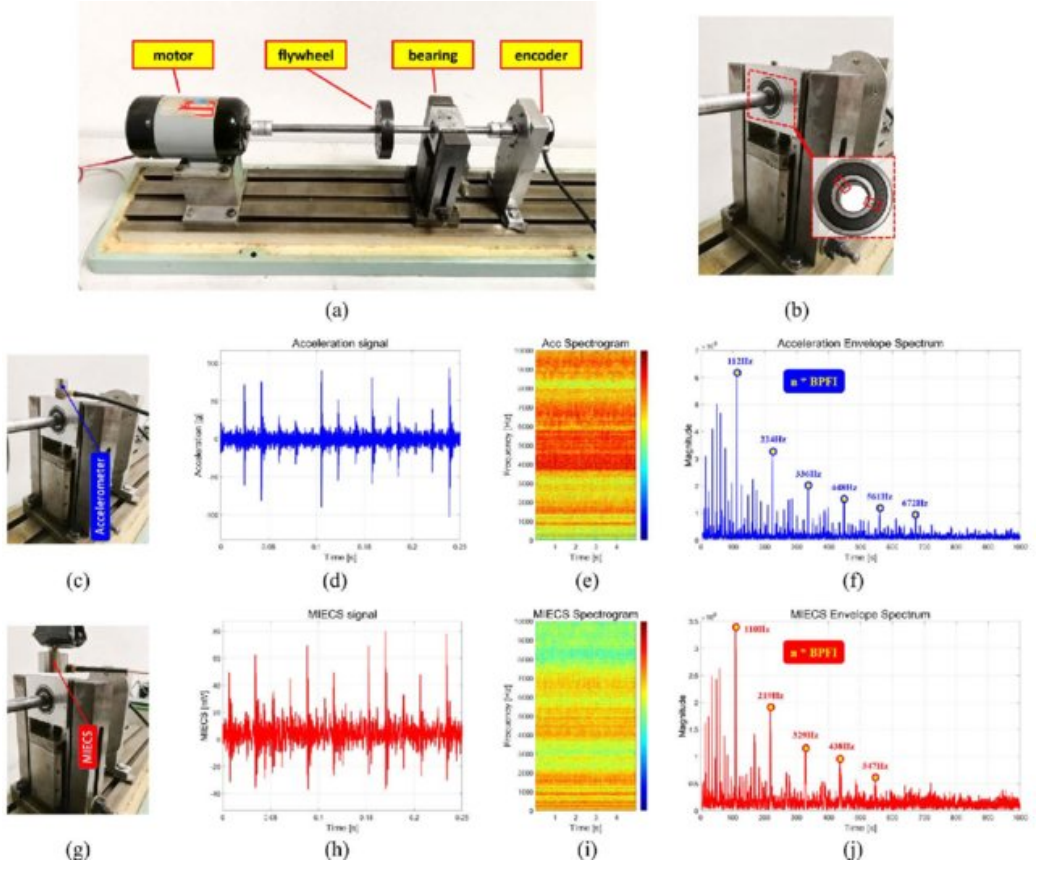
# Electroencephalography (EEG) brain monitoring




*An epilepsy seizure measured by EEG (top), corresponding spectrogram (middle), and power spectra at selected times (bottom).*

M.C. Ng et al., A Primer on EEG Spectrograms, Journal of Clinical Neurophysiology, 2021.

# Rotating machine monitoring




*(a) Rotor test bed. (b) Bearing support. (c)-(f): The time domain waveform, the spectrogram and the envelope spectrum of the signal acquired by the accelerometer. (g)-(j): The time domain waveform, the spectrogram and the envelope spectrum of the signal acquired by the MIECS.*

X. Xue et al., Motion induced eddy current sensor for non-intrusive vibration measurement, IEEE Sensors Journal, 2019.

# A bit of fun with Google Musiclab

https://musiclab.chromeexperiments.com/Spectrogram/

# A bit of fun with the CNRS

"Time-frequency analysis has a rich history since its first developments in the middle of the 20th century, at the interface of signal processing and harmonic analysis.

Time-frequency representations such as the spectrogram are routinely used in countless applications, ranging from gravitational astronomy to audio signal processing, through EEG or monitoring of rotating machinery."

# Discrete(-time) Fourier transform

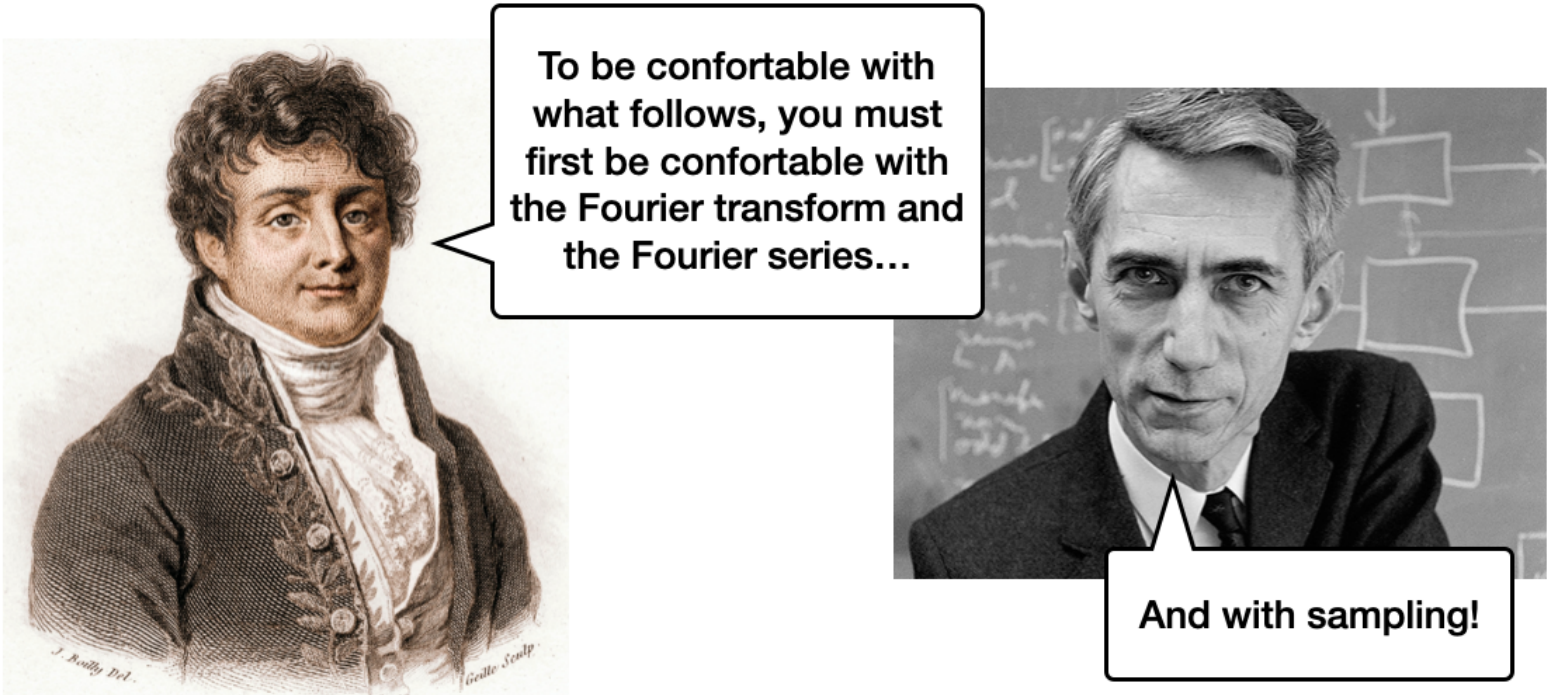
M. Bellanger, Traitement numérique du signal, Masson/CNET, Paris, 1990.

A.V. Oppenheim & W.S. Shaffer, Digital Signal Processing, Prentice Hall, NYC, 1975.

R.A. Robert & C.T. Mullis, Digital Signal Processing, Addison-Wesley, NYC, 1987.

# Discrete-time Fourier transform (DTFT)

- Prerequisite:




- Let $x(n) \in \mathbb{R}$ be a signal defined in the discrete time domain $n \in \mathbb{Z}$ such that $\sum_{n \in \mathbb{Z}} |x(n)| < \infty$
- $x(n)$ is generally the result of sampling a continuous signal $x_0(t)$, so that $x(n) = x_0(nT_s)$.

# Discrete-time Fourier transform (DTFT)

- The discrete-time Fourier transform is defined by

$$X(f) = \sum_{n \in \mathbb{Z}} x(n) e^{-j2\pi fn},$$

  where $f$ is called the normalized frequency (or digital frequency).

- By definition, the DTFT is periodic with period 1:

$$X(f+1) = \sum_{n \in \mathbb{Z}} x(n) e^{-j2\pi (f+1)n} = \sum_{n \in \mathbb{Z}} x(n) e^{-j2\pi fn} e^{-j2\pi n} = \sum_{n \in \mathbb{Z}} x(n) e^{-j2\pi fn} = X(f).$$

  We thus limit its representation to the interval $f \in [0, 1[$ or $f \in ]-0.5, 0.5]$ (+ we exploit the symmetry, see later).

# DTFT ("alternate definition")

- Reminder: The "ideally sampled" signal and its Fourier transform:

$$x_s(t) = x_0(t) \sum_{n\in\mathbb{Z}} \delta(t - nT_s) = \sum_{n\in\mathbb{Z}} x_0(nT_s)\delta(t - nT_s) \overset{\text{FT}}{\longleftrightarrow} X_s(\nu) = \sum_{n\in\mathbb{Z}} x_0(nT_s)e^{-j2\pi\nu nT_s},$$

  where $\nu$ is the "physical" (analog) frequency (in Hz).

- We thus have $X(f) = X_s(fF_s)$ with $f = \frac{\nu}{F_s}$, i.e., $X(f)$ is a scaled version of $X_s(\nu)$ along the frequency axis.

- Remind that we also have $X_s(\nu) = F_s \sum_{k\in\mathbb{Z}} X_0(\nu - kF_s)$, where $X_0(\nu) = \text{FT}[x_0(t)]$, which is another way to highlight the periodicity of $X(f)$.

- $X(f)$ is studied for $f \in ]-0.5, 0.5]$, whereas $X_s(\nu)$ is studied for $\nu \in ]-\frac{F_s}{2}, \frac{F_s}{2}]$.

# Some properties

| | Discrete-time domain | DTFT domain |
|---|---|---|
| Linearity | $ax(n) + by(n)$ | $aX(f) + bY(f)$ |
| Time shift | $x(n - n_0)$ | $X(f)e^{-j2\pi f n_0}$ |
| Frequency shift | $x(n)e^{j2\pi f_0 n}$ | $X(f - f_0)$ |
| Convolution | $[x \star y](n) = \sum_{k\in\mathbb{Z}} x(k)y(n-k)$ | $X(f)Y(f)$ |
| Hermitian sym. | $x(n) \in \mathbb{R}$ | $X(f) = X^*(-f)$ |

Thanks to the Hermitian symmetry, the DTFT spectrum of real-valued signals is often considered only for $f \in [0, 0.5]$.

# Inverse discrete-time Fourier transform (IDTFT)

- The inverse discrete-time Fourier transform is defined by:

$$x(n) = \int_{-1/2}^{1/2} X(f) e^{+j2\pi f n} df.$$

- For real-valued signals, the Hermitian symmetry ensures that the resulting "resynthesized" signal is real-valued. Two complex sinusoids at frequency $f$ and $-f$ combine into a real sinusoid at frequency $f$.

- If $x(n)$ contains a high-power component at frequency $f$, the value of $|X(f)|$ will be large.

# Signals of finite length - Frequency sampling

- In practice, we work with signals of finite length: $x(n) \neq 0$ only for $n \in \{0, ..., N-1\}$,

  and we have: $\sum_{n \in \mathbb{Z}} |x(n)| = \sum_{n=0}^{N-1} |x(n)| < \infty$.

  $\{0, ..., N-1\}$ is called the support of the signal.

- In practice, we cannot compute a DTFT for an infinite (continuous) set of frequency values. We thus also work with a finite subset of frequencies. Let us fix a number of frequencies $K \in \mathbb{N}$ and define

$$f_k = \frac{k}{K} \in [0, 1[, \qquad k \in \{0, ..., K-1\}$$

  Let us then sample the DTFT, in order to obtain the discrete Fourier transform (DFT).

# Discrete Fourier transform (DFT)

Let $x(n) \in \mathbb{R}$, $n \in \mathbb{Z}$, be a signal of finite support $\{0, ..., N-1\}$.

- The discrete Fourier transform (DFT) of order $K \in \mathbb{N}$ is defined by:

$$X_K(k) = X\left(\frac{k}{K}\right) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{kn}{K}}, \qquad k \in \{0, ..., K-1\}.$$

- We also have: $\nu_k = k\frac{F_s}{K} \in [0, F_s[$ and $X_K(k) = X_s(k\frac{F_s}{K})$.

  $\frac{F_s}{K}$ is the sampling period in the (physical) frequency domain and thus $\frac{K}{F_s}$ is the sampling frequency ($\frac{1}{K}$ is the sampling period in the normalized frequency domain).

- If we don't want to loose information, the frequency-domain sampling must verify the Shannon condition, i.e. $\frac{K}{F_s} \geq NT_s \Leftrightarrow K \geq N$.

  ($NT_s$ is the "width" of the time-domain signal, which is periodized due to the frequency-domain sampling, and we want to avoid aliasing/overlap in the time domain.)

# Discrete Fourier transform (DFT)

- In practice, we have two cases:
  - We set $K = N$. This ensures the minimal number of computations for representing the spectrum without losing information in the Shannon sense.
  - We "set" $K > N$ by artificially adding $K - N$ zeros to the $N$-point samples sequence, so that the DFT algorithm computes it on $K$ values. This is called zero-padding. Zero-padding is often applied (automatically) when using fast Fourier transform (FFT) algorithms that require $K$ being a power of two (more details later).
- Note: $K$ is often "renamed" $N$ in any case (i.e., $N$ is the DFT size in the frequency domain with or without zero-padding), so that the definition of the DFT in the literature is generally:

$$X_N(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{kn}{N}}, \quad k \in \{0, ..., N-1\}.$$

We keep this formulation in the following.

# Some properties

| | Discrete-time domain | DFT domain |
|---|---|---|
| Linearity | $ax(n) + by(n)$ | $aX_N(k) + bY_N(k)$ |
| Time shift | $\tilde{x}(n - n_0)$ | $X_N(k)e^{-j2\pi\frac{kn_0}{N}}$ |
| Frequency shift | $x(n)e^{j2\pi\frac{k_0 n}{N}}$ | $\tilde{X}_N(k - k_0)$ |
| Circular convol. | $[x \circledast y](n) = \sum_{k=0}^{N-1} \tilde{x}(k)\tilde{y}(n-k)$ | $X_N(k)Y_N(k)$ |
| Hermitian sym. | $x(n) \in \mathbb{R}$ | $X_N(k) = X_N^*(N-k)$ |

Note: $\tilde{x}(n)$ denotes the $N$-periodized version of $x(n)$.

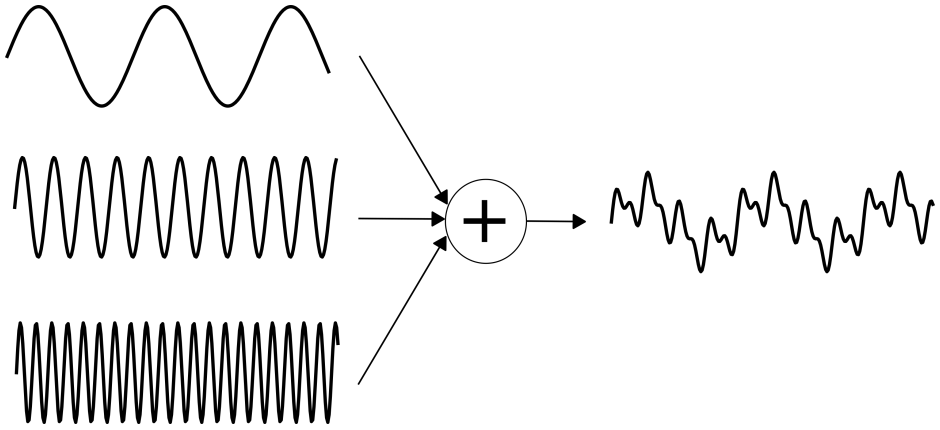
Thanks to the Hermitian symmetry, the DFT spectrum of real-valued signals is often studied only for $k \in [0, \frac{N}{2}]$.

# Inverse discrete Fourier transform (IDFT)

- The inverse discrete Fourier transform is defined by:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_N(k) e^{+j2\pi \frac{kn}{N}},$$

$$n \in \{0, ..., N-1\}.$$



- The time-domain signal is represented as a linear combination of complex sinusoids
- The Hermitian symmetry $X_N(N-k) = X_N^*(k)$ ensures that the resulting signal is real-valued. Two complex sinusoids at frequency $+k/N$ and $-k/N$ (actually at frequency bins $k$ and $N-k$) combine into a real sinusoid at frequency $k/N$.
- If $x(n)$ contains a component at frequency $k/N$ (or more precisely, if $x_0(t)$ contains a component within the $k$-th channel of width $F_s/N$ around the frequency $kF_s/N$), the modulus of the corresponding DFT coefficient $|X_N(k)|$ will be large.

## Inverse discrete Fourier transform (IDFT)

- The IDFT formula shows that $x(n)$ is implicitly $N$-periodized by the sampling in the frequency domain. Therefore, from this point of view, both $x(n)$ and $X_N(k)$ are $N$-periodic!

# Alternative definition by normalizing

Let $x(n) \in \mathbb{R}$, $n \in \mathbb{Z}$, be a signal of finite support $\{0, ..., N' - 1\}$ with $N' \leq N$.

- The discrete Fourier transform (DFT) of order $N \in \mathbb{N}$ is defined by:

$$X_N(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{nk}{N}}, \quad k \in \{0, ..., N-1\}.$$

- The inverse discrete Fourier transform is defined by:

$$x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_N(k) e^{+j2\pi \frac{nk}{N}}, \quad n \in \{0, ..., N-1\}.$$

# Geometric interpretation

- Let $\mathbf{x} \in \mathbb{R}^N$ be the vector of entry $x(n)$, $n \in \{0, ..., N-1\}$.
- Let $\mathbf{u}_k \in \mathbb{C}^N$ be the vector of entry $\frac{1}{\sqrt{N}} e^{+j2\pi \frac{nk}{N}}$, $n \in \{0, ..., N-1\}$, i.e., $\mathbf{u}_k$ is ($k$ periods of) a complex sinusoid at frequency $k/N$.
- For each frequency bin $k$, the DFT can be expressed as an inner product (on the complex vector space $\mathbb{C}^N$) between $\mathbf{x}$ and $\mathbf{u}_k$:

$$X_N(k) = \langle \mathbf{x}, \mathbf{u}_k \rangle = \mathbf{u}_k^H \mathbf{x}, \qquad k \in \{0, ..., N-1\},$$

  where $\cdot^H$ denotes Hermitian transpose (transpose + complex conjugate).
- The modulus of the inner product $|X_N(k)| = |\langle \mathbf{x}, \mathbf{u}_k \rangle| = |\mathbf{u}_k^H \mathbf{x}|$ can be interpreted as a measure of similarity between the two vectors. Again, if $x(n)$ "contains" a complex sinusoid of frequency $k/N$, the modulus of the corresponding DFT coefficient $|X_N(k)|$ will be high.

# Matrix formulation

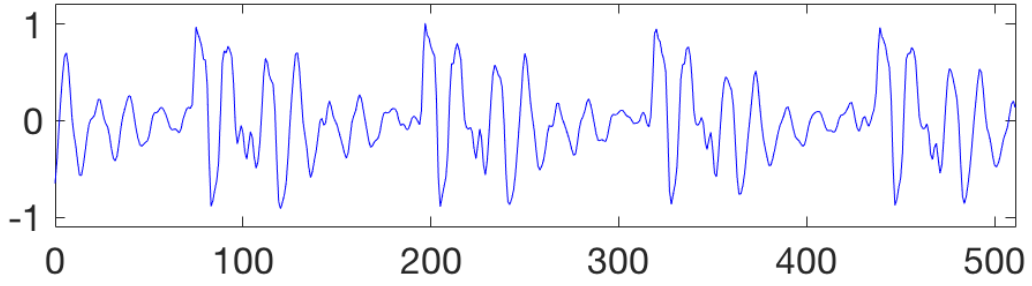
- Let $\mathbf{X} \in \mathbb{C}^N$ be the vector of entry $X_N(k),\ k \in \{0, ..., N-1\}$.
- Let $\mathbf{U} \in \mathbb{C}^{N \times N}$ be the DFT matrix where each row of index $k \in \{0, ..., N-1\}$ is defined by $\mathbf{u}_k^H$.
- Then the DFT can be expressed as: $\mathbf{X} = \mathbf{U}\mathbf{x}$.
- It can be shown that the vectors $\{\mathbf{u}_k\}_{k=0}^{N-1}$ form an othonormal basis of $\mathbb{C}^N$: $\mathbf{U}^H\mathbf{U} = \mathbf{I}_N$.
- Therefore, the inverse DFT can be written as: $\mathbf{U}^H\mathbf{X} = \mathbf{U}^H\mathbf{U}\mathbf{x} = \mathbf{x}$.
- In summary: the DFT/IDFT is just a change of coordinate system.

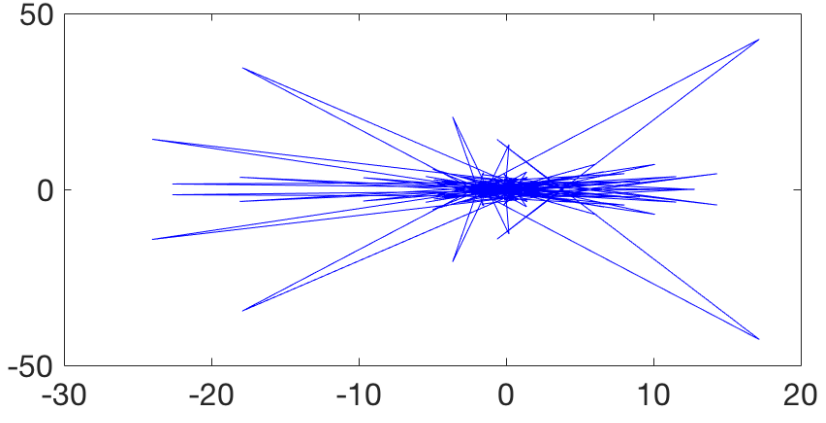
  The DFT is given by the projection of the signal onto the set of complex sinusoids $\{\mathbf{u}_k\}_{k=0}^{N-1}$ and the inverse DFT is the reconstruction of the signal as a superposition of these projections.

$$\mathbf{x} = \sum_{k=0}^{N-1} X_N(k)\mathbf{u}_k = \sum_{k=0}^{N-1} \langle \mathbf{x}, \mathbf{u}_k \rangle \mathbf{u}_k.$$

# Complex spectrum

- For general (unconstrained) real-valued signals, the D(T)FT is complex-valued. It is called the complex spectrum of the signal, or loosely speaking simply the spectrum.
- In Matlab, if you type: "s = audioread('a.wav'); plot(s); S = fft(s); plot(S);", you get:



# Magnitude, phase and power spectra

- The modulus of the D(T)FT is called the magnitude spectrum.
- The argument of the D(T)FT is called the phase spectrum.
- The squared modulus of the D(T)FT is called the power spectrum (or power spectral density, remember the dual definition of signal power in the time and frequency domains).

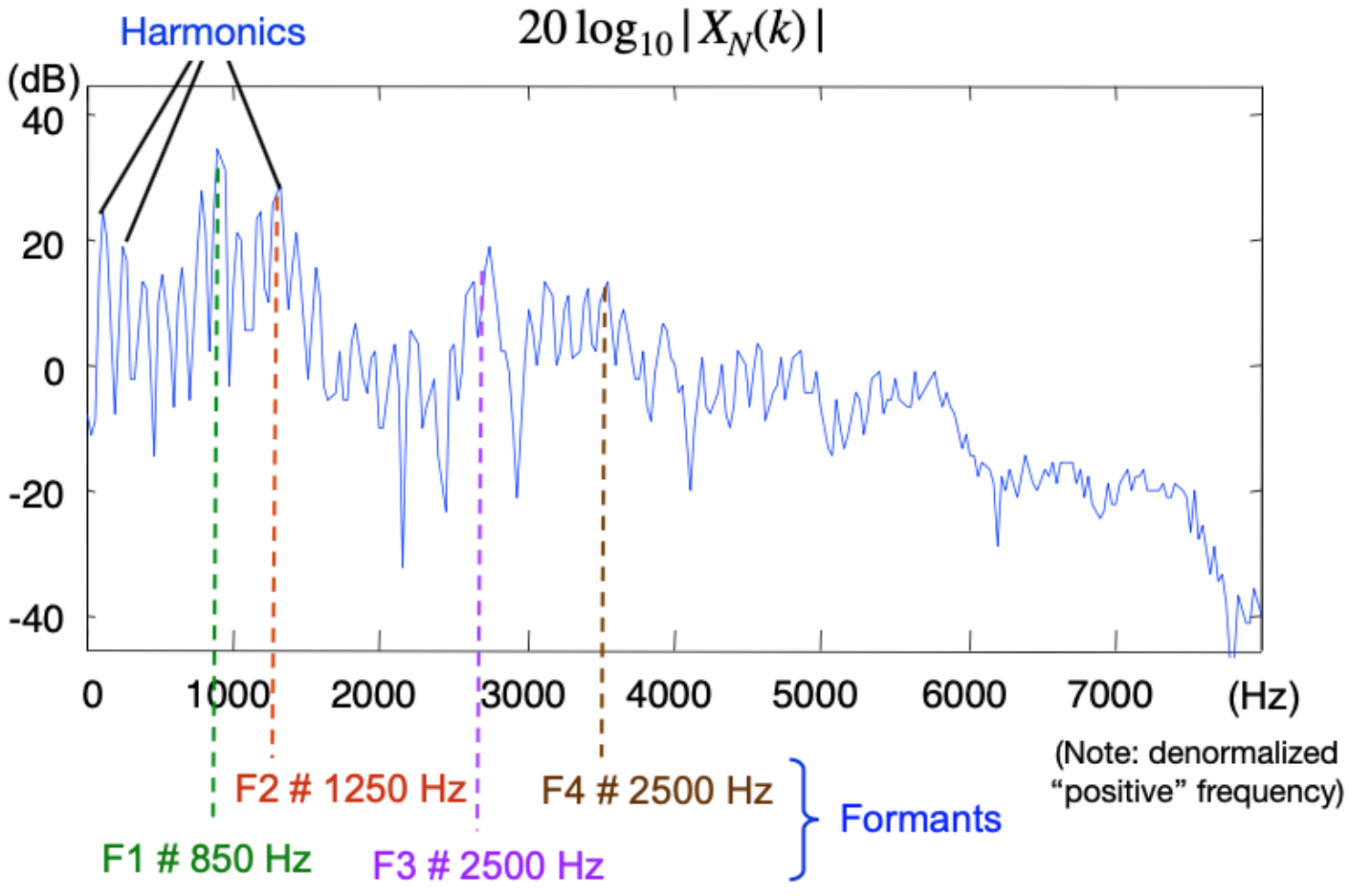
> The complex spectrum is generally visualized via the magnitude or power spectrum.

(it can be useful to also visualize the phase spectrum in some applications, but in general, for audio signals, the phase spectrum is much less informative than the magnitude/power spectrum.)

- When displayed on a log scale, the magnitude and the power spectra are almost equivalent. The dB scale is generally used.

# An example of magnitude/power spectrum of a real-world signal

32 ms of the wowel /a/ pronounced by a male speaker, sampled at 16 kHz.



# Time-domain support vs. frequency-domain resolution

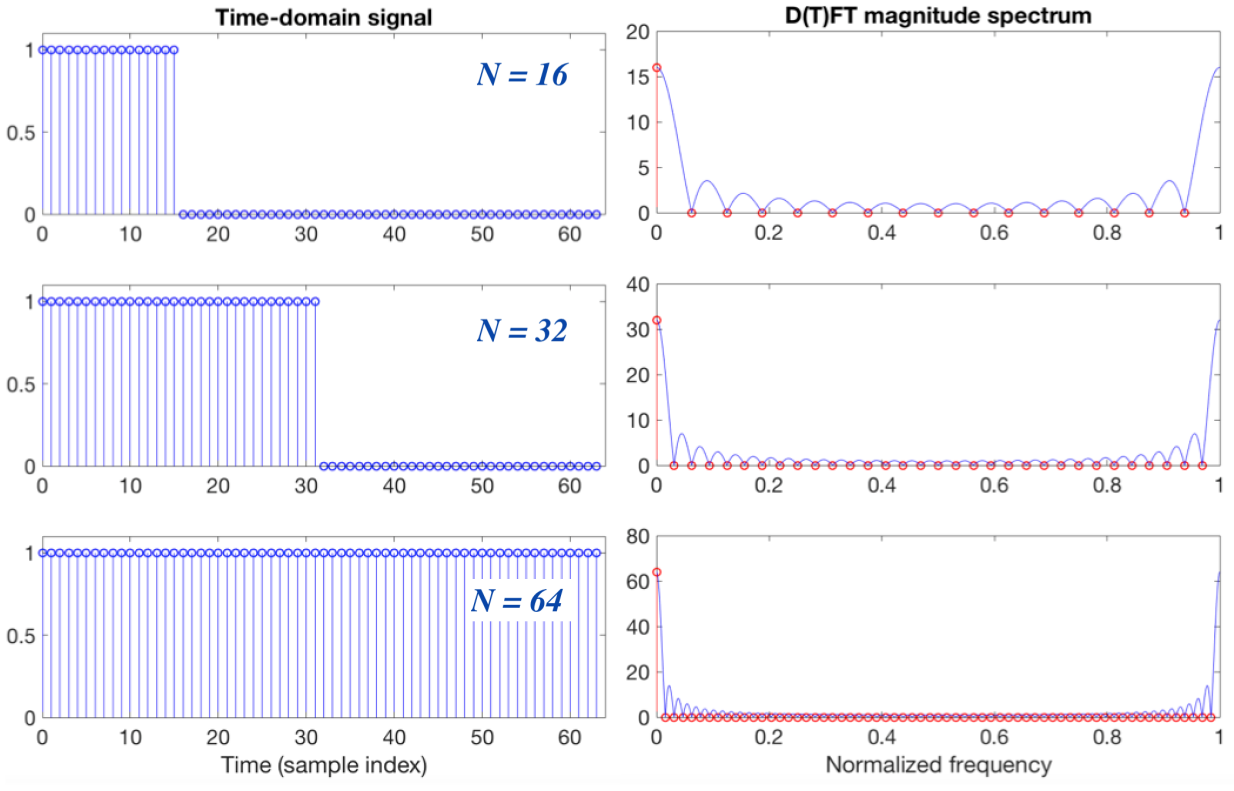
- Let us consider the finite-length constant signal defined by

$$w(n) = \begin{cases} 1 & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise,} \end{cases}$$

- Its DTFT is a periodized sine cardinal (sinc) given by:

$$W(f) = \frac{\sin(\pi f N)}{\sin(\pi f)} e^{-j\pi f(N-1)}$$

with principal lobe width = $2/N$.




The shorter the time-domain signal, the larger the principal lobe, the lower the frequency-domain resolution.

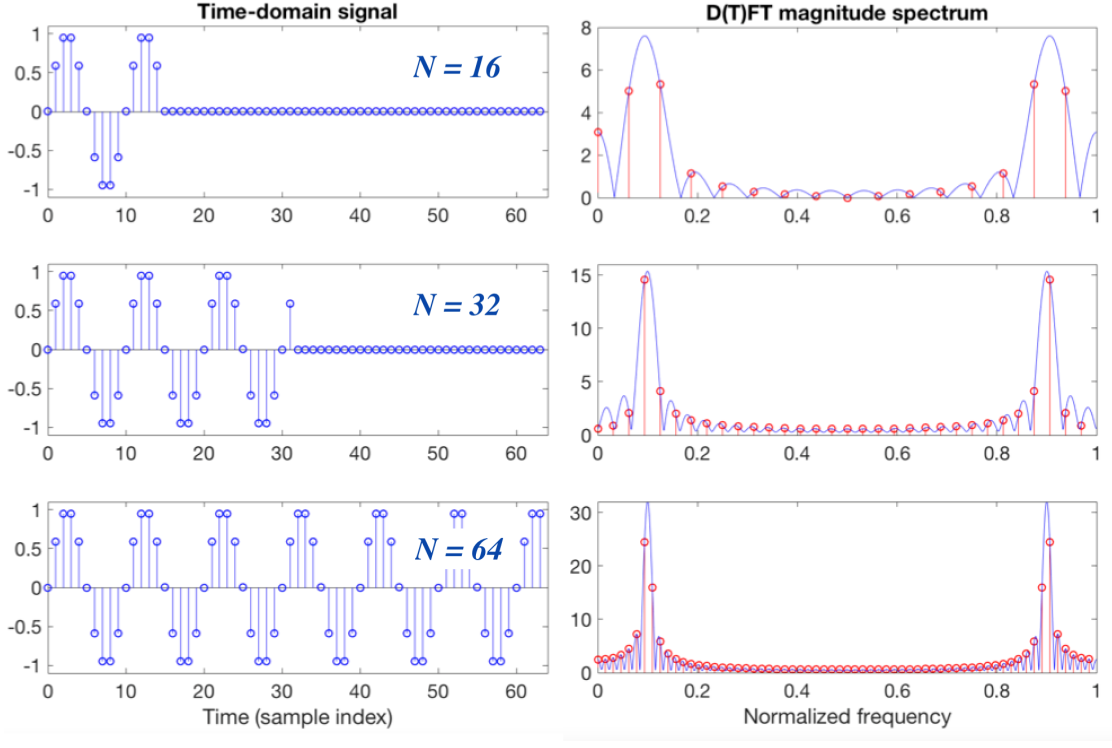
Question: How is the blue curve computed?

# Time-domain support vs. frequency-domain resolution

- Let us consider the finite-length pure tone: $x(n) = \begin{cases} \sin(2\pi f_0 n) & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise.} \end{cases}$
- The DTFT of $x(n)$ is the convolution of a pair of Dirac at $\pm f_0$ with $W(f)$.

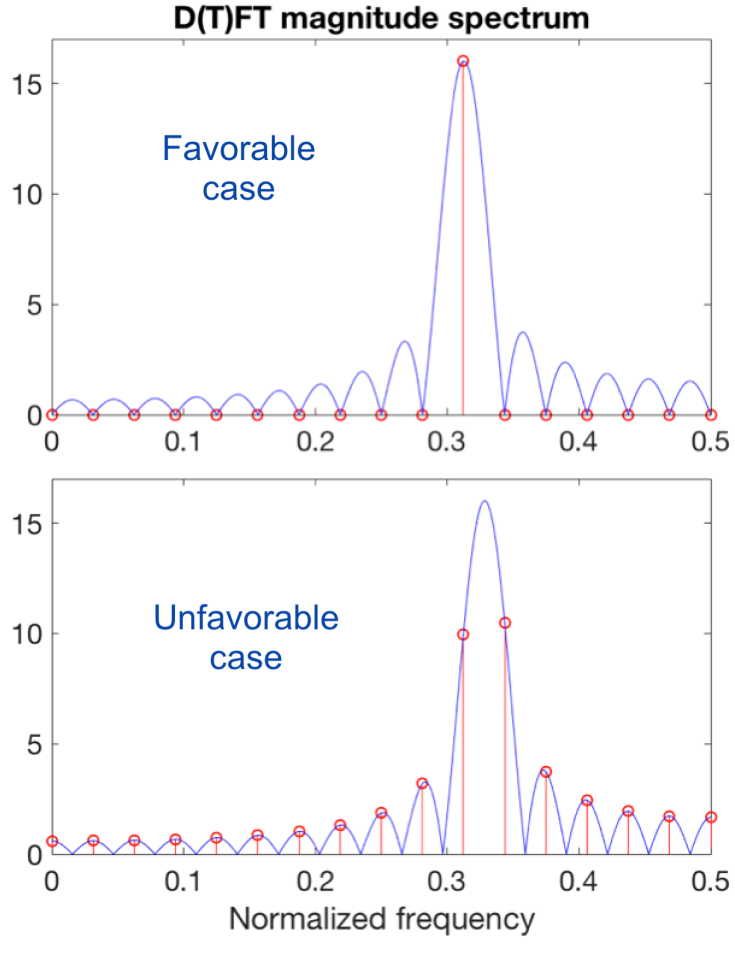
- Two principal lobes centered around $f_0$ and $-f_0$ (or $1 - f_0$) (here $f_0 = 0.1$).
- The same principle as for $W(f)$ rules the main lobe width.

  The more periods we observe, the better the resolution = the more certain we are about the pure tone frequency, as measured from the DFT points.



# Influence of sampling depending on $f_0$ value

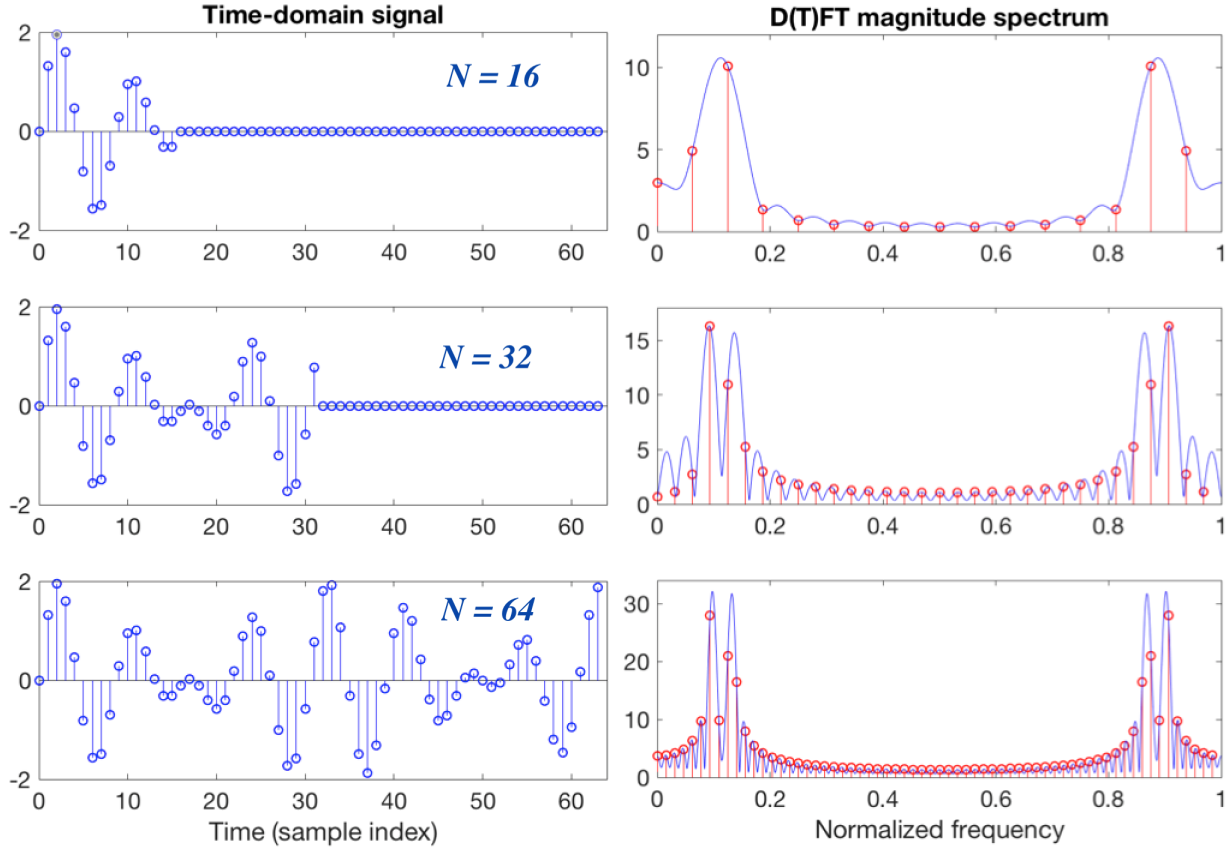
- Favorable case: $f_0 = \frac{J}{N}$ with $J$ being integer, i.e., $f_0$ is a multiple of the sampling period (in the normalized frequency domain).
- Unfavorable case: $f_0 = \frac{J+0.5}{N}$.
- In practice, we don't control the ratio between $f_0$ and $1/N$, hence we don't control the "quality" of the sampled spectral peaks. We may want to use zero-padding to increase the frequency-domain sampling (though not increasing the resolution).



# D(T)FT of the sum of pure tones

$$x(n) = \begin{cases} \sin(2\pi f_0 n) + \sin(2\pi f_1 n) & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad \text{where } f_0 = 0.1,\ f_1 = 0.13$$

- For $N = 16$, the resolution is not sufficient to separate the two tones for both the DFT and DTFT.
- For $N = 32$, the resolution is sufficient to separate the two tones for the DTFT but not for the DFT. This may benefit from zero-padding.
- For $N = 64$, it's OK to separate the two tones for both the DTFT and DFT, though the second tone is poorly represented by the DFT.



# Link with Fourier series

- A perfectly $T_0$-periodic continuous (real-valued) signal $x_0(t), t \in \mathbb{R}$, decomposes into a Fourier series, i.e., a sum of sinusoids at multiple of the fundamental frequency $\nu_0 = 1/T_0$. The sinusoidal components are called harmonics.

- There are several expressions of the Fourier series. In audio processing, we favour the following one, a.k.a. the harmonic model:

$$x_0(t) = \sum_{i=0}^{+\infty} A_i \cos(2\pi i \nu_0 t + \phi_i), \quad t \in \mathbb{R},$$

  with $A_i \in \mathbb{R}^+$ and $\phi_i \in [0, 2\pi[$.

- The real-world digital version is bandlimited, sampled and of finite length:

$$x(n) = \sum_{i=0}^{I} A_i \cos(2\pi i f_0 n + \phi_i), \quad \text{for } n \in \{0, ..., N-1\},$$

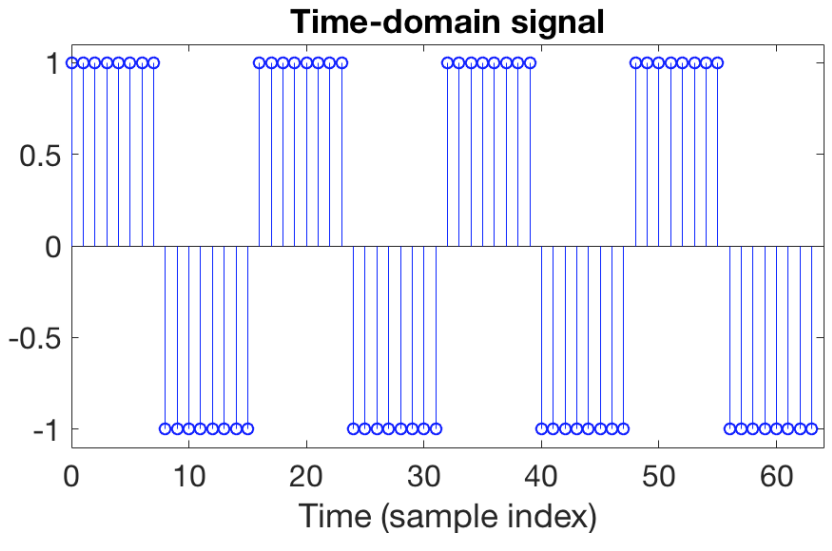
  and $0$ otherwise, with $I\nu_0 < F_s/2 \Leftrightarrow I f_0 < 0.5$.

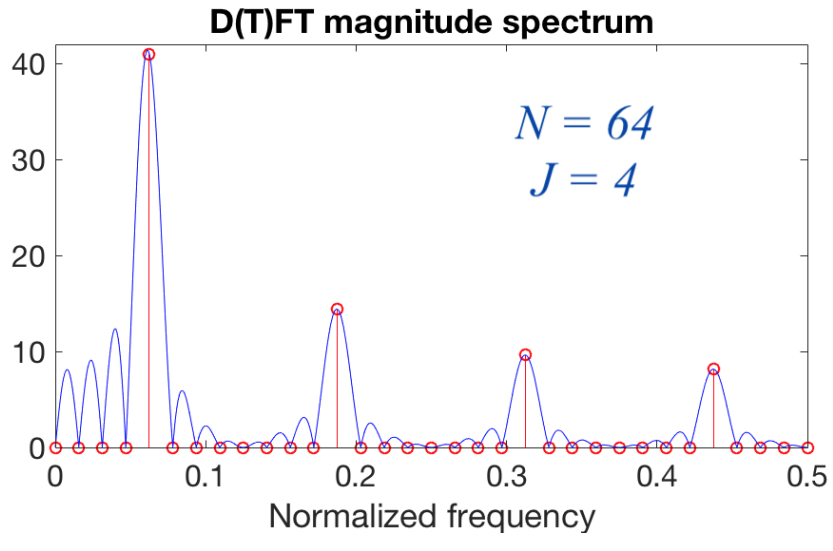
# Link with Fourier series

- If $f_0 = \frac{J}{N}$ with $J$ being an integer, the "favorable case" of frequency-domain sampling applies to the lobe peak at $f_0$ and to all harmonics at multiples of $f_0$.

- In that case, the DFT coefficients correspond to the Fourier series coefficients of $x_0(t)$:

$$X_N(k) = \begin{cases} C_i & \text{if } k = iJ \\ 0 & \text{otherwise} \end{cases}$$



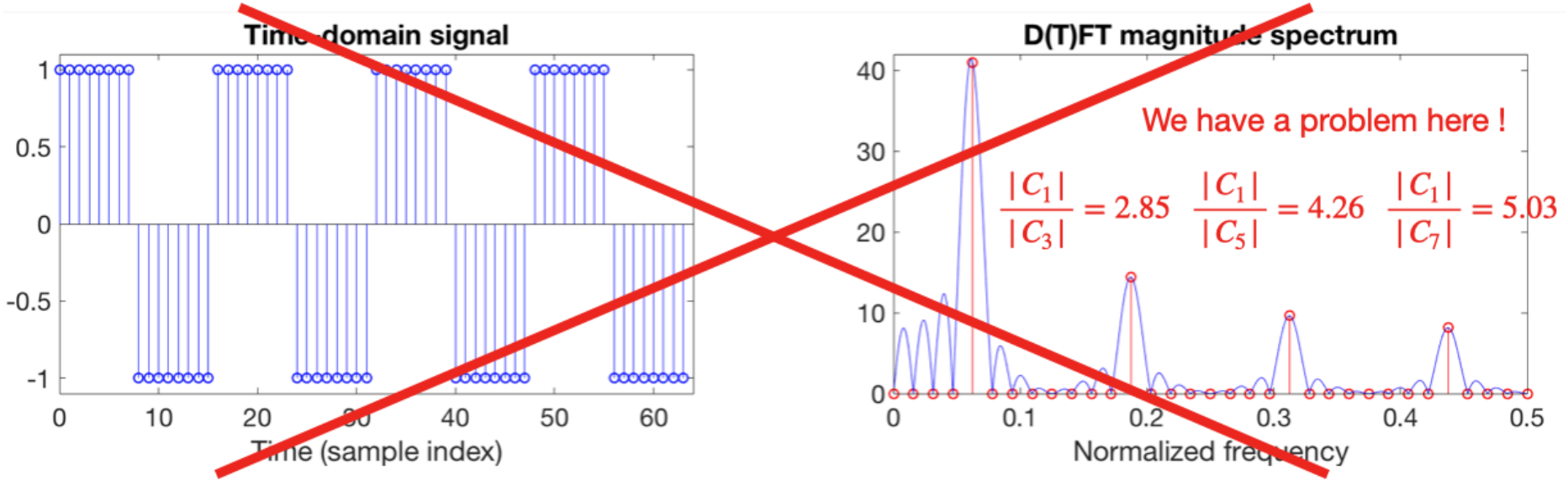



# Link with Fourier series

- If $f_0 = \frac{J}{N}$ with $J$ being an integer, the "favorable case" of frequency-domain sampling applies to the lobe peak at $f_0$ and to all harmonics at multiples of $f_0$.

- In that case, the DFT coefficients correspond to the Fourier series coefficients of $x_0(t)$:

$$X_N(k) = \begin{cases} C_i & \text{if } k = iJ \\ 0 & \text{otherwise} \end{cases}$$

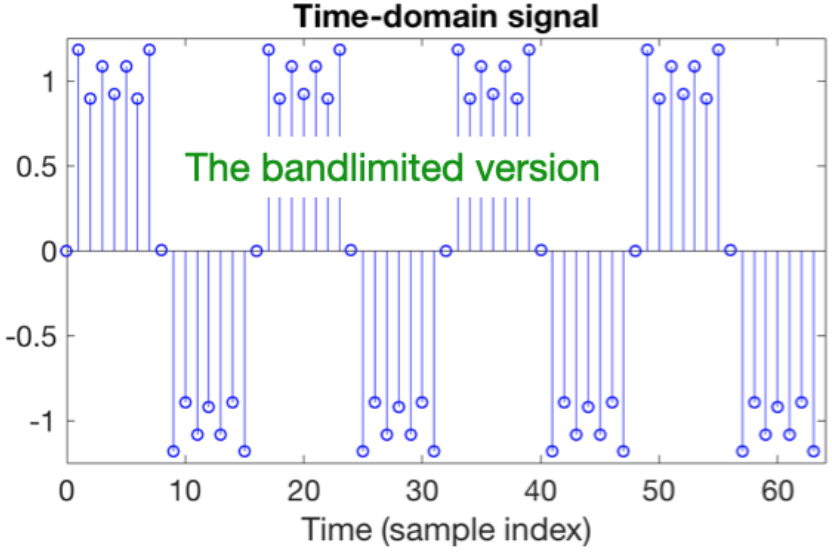


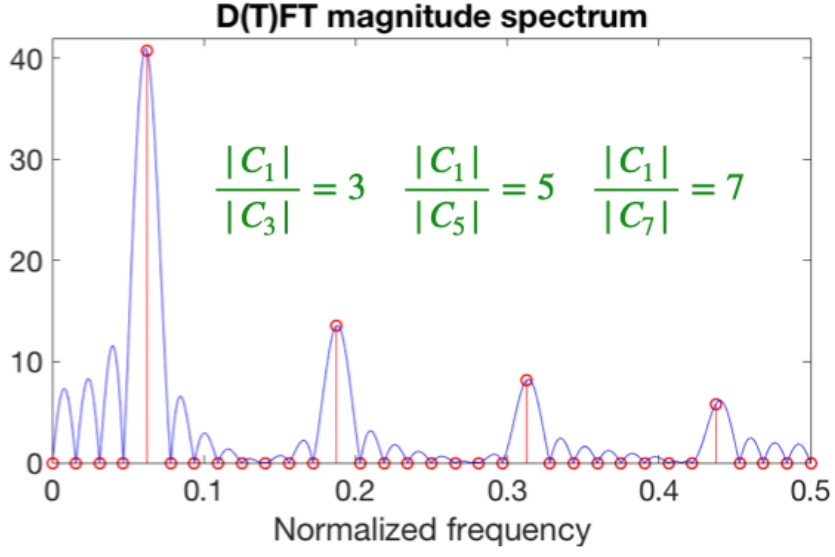
# Link with Fourier series

- If $f_0 = \frac{J}{N}$ with $J$ being an integer, the "favorable case" of frequency-domain sampling applies to the lobe peak at $f_0$ and to all harmonics at multiples of $f_0$.

- In that case, the DFT coefficients correspond to the Fourier series coefficients of $x_0(t)$:

$$X_N(k) = \begin{cases} C_i & \text{if } k = iJ \\ 0 & \text{otherwise} \end{cases}$$

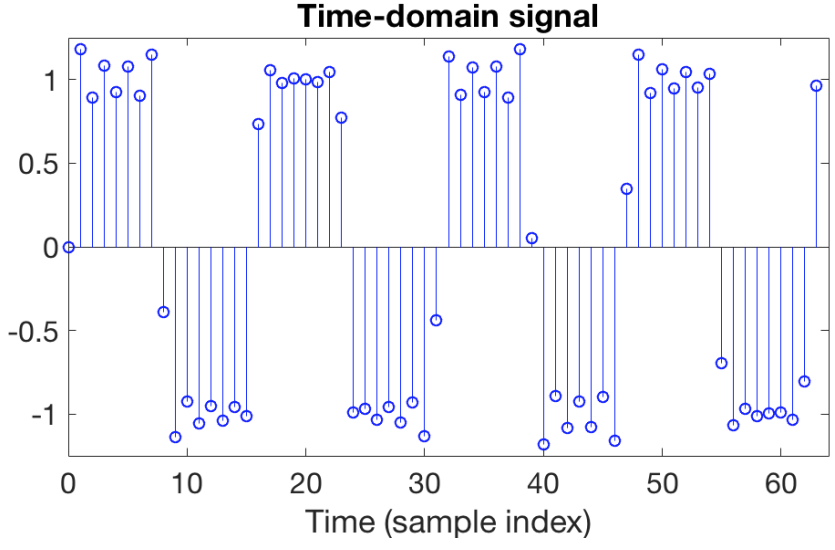


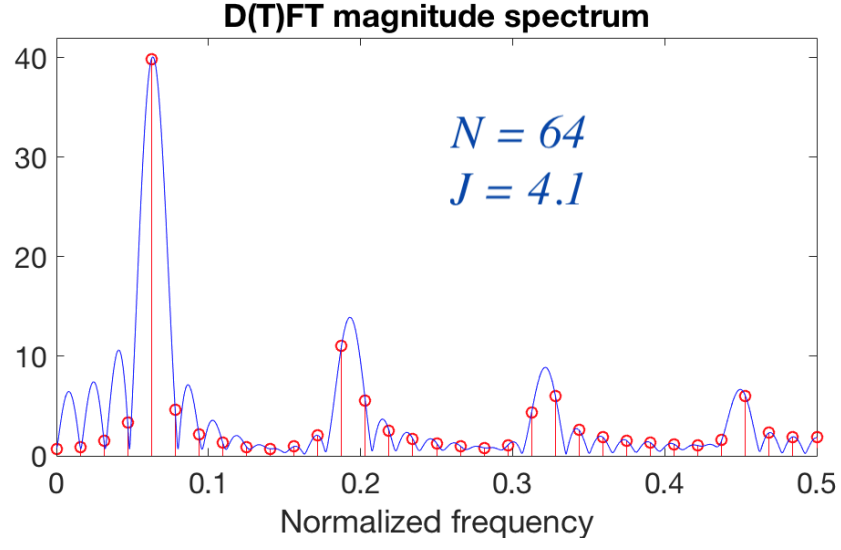



# Link with Fourier series

- If the above condition is not fullfilled, then sampling of the DTFT happens at arbitrary positions, more or less favorable.

  In general, there are peaks in the DFT spectrum near the harmonics, but we must keep in mind that they generally do not correspond exactly to the peaks of the underlying DTFT lobes.



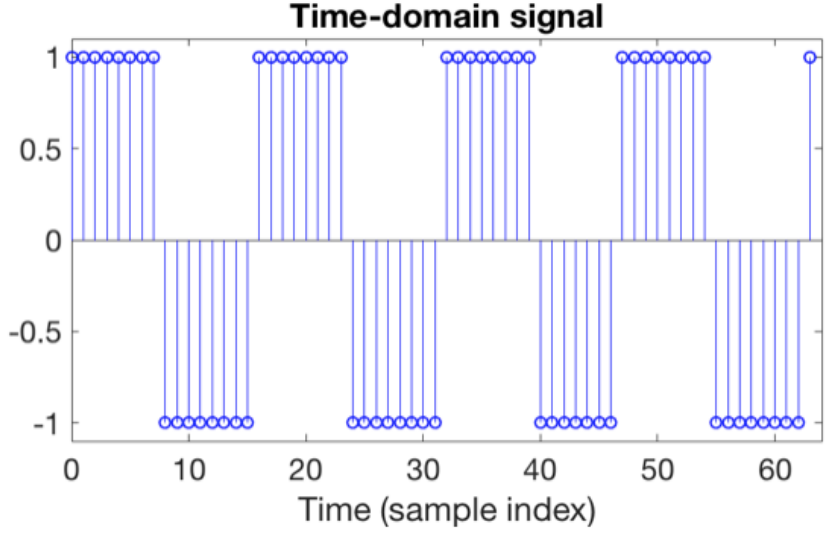



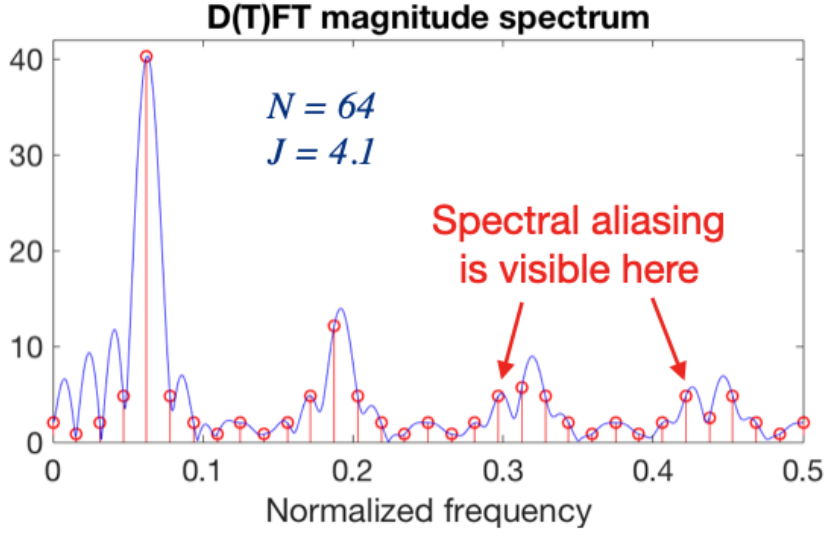
# Link with Fourier series

- If the above condition is not fullfilled, then sampling of the DTFT happens at arbitrary positions, more or less favorable.

  In general, there are peaks in the DFT spectrum near the harmonics, but we must keep in mind that they generally do not correspond exactly to the peaks of the underlying DTFT lobes.

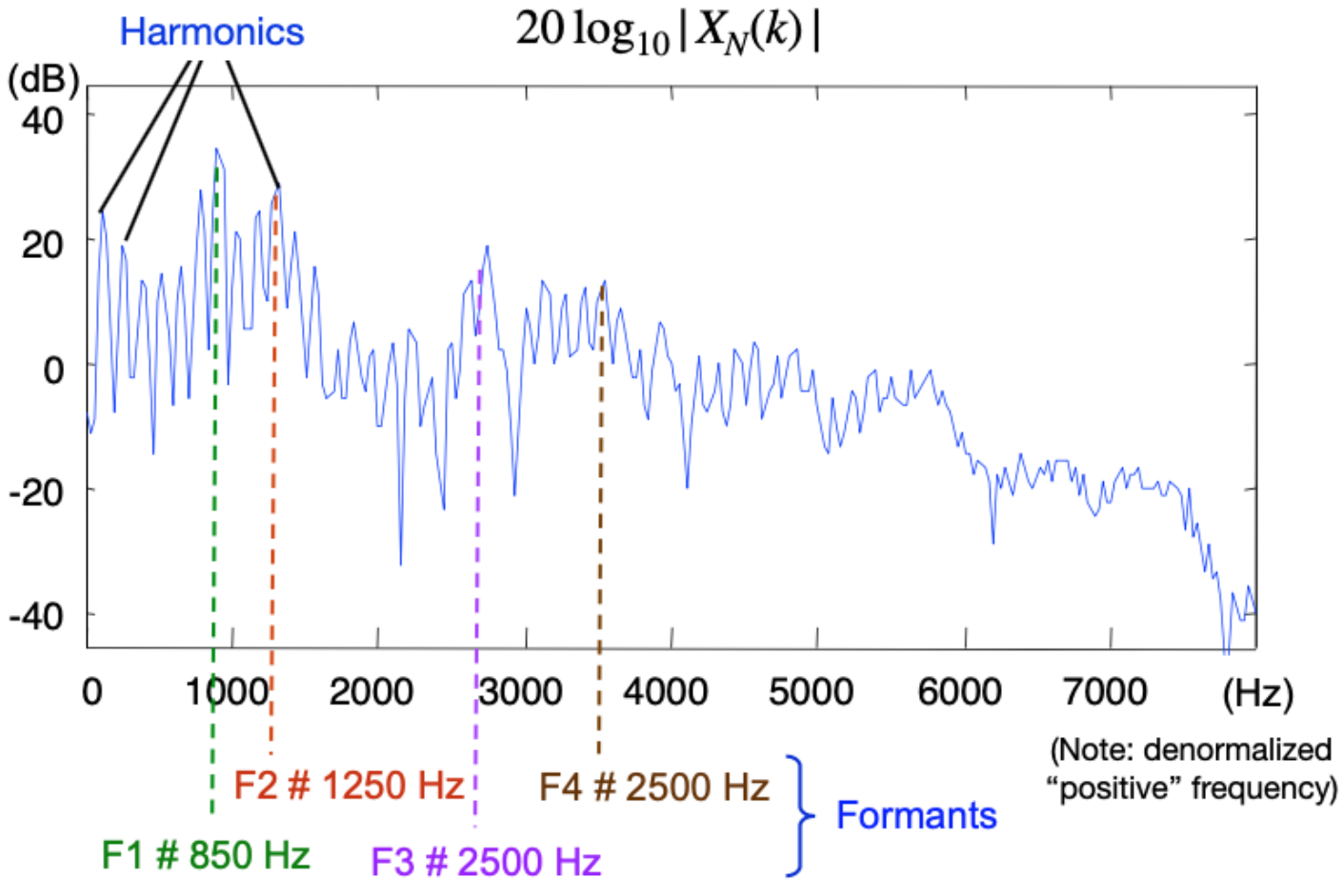
  *Just fo the fun, let us see the combination of unlimited band signal and "non-synchronous" sampling:*



# An example of magnitude/power spectrum of a real-world signal

32 ms of the wowel /a/ pronounced by a male speaker, sampled at 16 kHz.



# Window function

- The use of a rectangular function to select a portion of signal with limited support has a large effect on the spectrum shape. It "transforms" a Dirac into a (periodic) sinc.
- We may want to use another window. This may provide a better control of the spectral peaks shape and resolution.
- In a general manner, these window functions have a smooth bell shape.
- The DFT is rewritten:

$$X_N(k) = \sum_{n=0}^{N-1} x(n) w_a(n) e^{-j2\pi \frac{kn}{N}}, \quad k \in \{0, ..., N-1\},$$

  where $w_a(n)$ denotes the analysis window, of support $\{0, ..., N-1\}$.

- Note: Windowing can be combined with zero-padding. In general, if zero-padding is used, it is better to apply it after windowing.

# Some usual window functions

Rectangular

$$w_a(n) = 1$$

Hamming

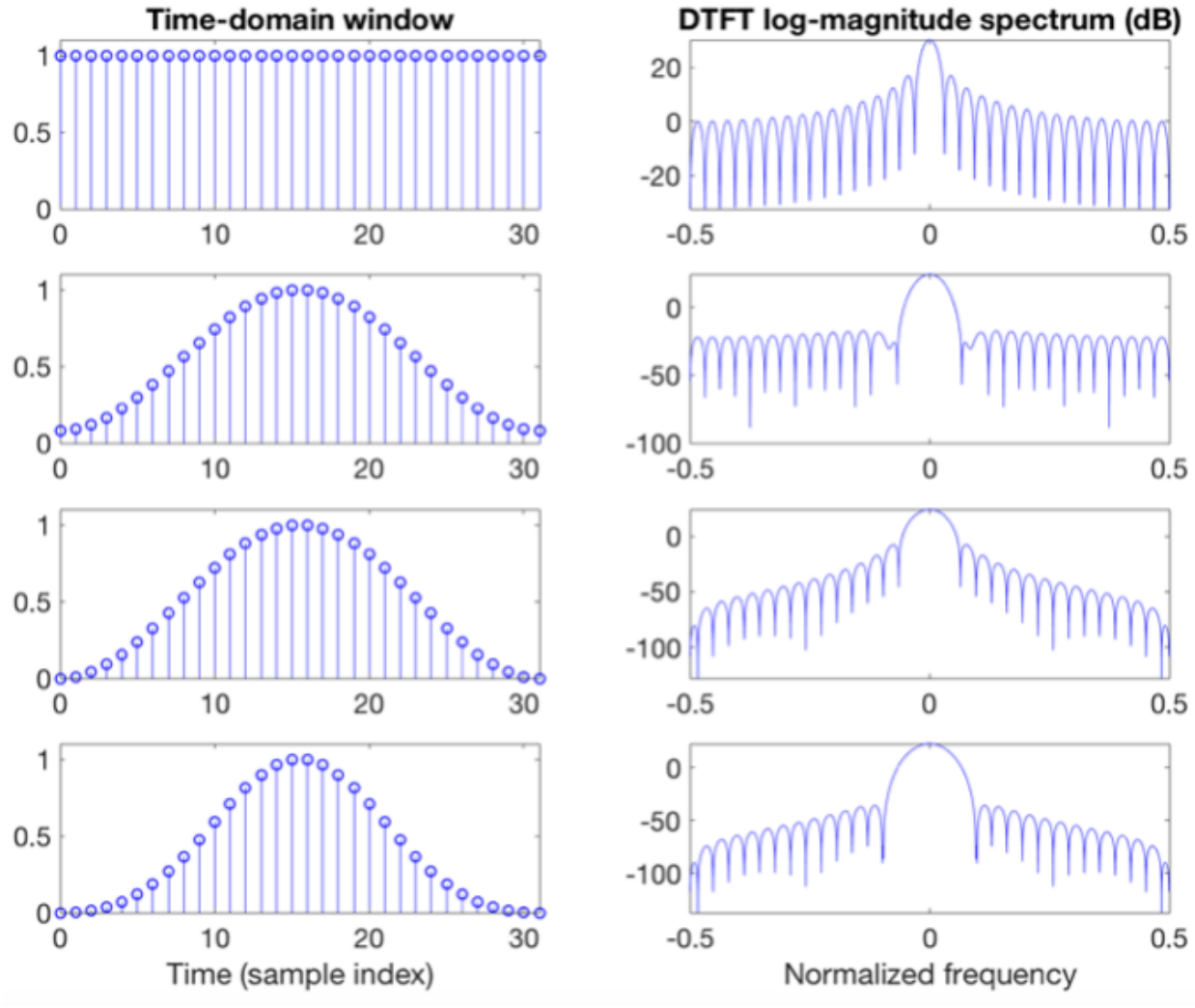
$$w_a(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N-1}\right)$$

Hann

$$w_a(n) = 0.5\left(1 - \cos\left(2\pi\frac{n}{N-1}\right)\right)$$

Blackman

$$w_a(n) = 0.42 - 0.5\cos\left(2\pi\frac{n}{N-1}\right) + 0.08\cos\left(4\pi\frac{n}{N-1}\right)$$



# Window functions properties

| Window | Main lobe width (Hz) | First secondary lobe attenuation (dB) | Secondary lobes attenuation speed |
|---|---|---|---|
| Rectangular | $2\frac{F_S}{N}$ | $-13$ | Slow |
| Hann | $4\frac{F_S}{N}$ | $-31$ | Fast |
| Hamming | $4\frac{F_S}{N}$ | $-41$ | Slow |
| Blackman | $6\frac{F_S}{N}$ | $-57$ | Fast |

# Fast Fourier transform (FFT)

- FFT = a family of algorithms for fast computation of the DFT
- Example: The butterfly FFT algorithm
- Requires that $N$ be a power of $2$: $N = 2^p$
- Computation cost evaluated in number of multiplication-accumulation (MAC) operations:
  - "Naive" DFT computation: $N^2$ MACs
  - butterfly-type FFT: $N \log_2(p)$ MACs
- Example: $N = 1024$; $N^2 \approx 10^6$ MACs; $N \log_2(p) \approx 10000$ MACs.

# Back to music signals: Pitch and timbre

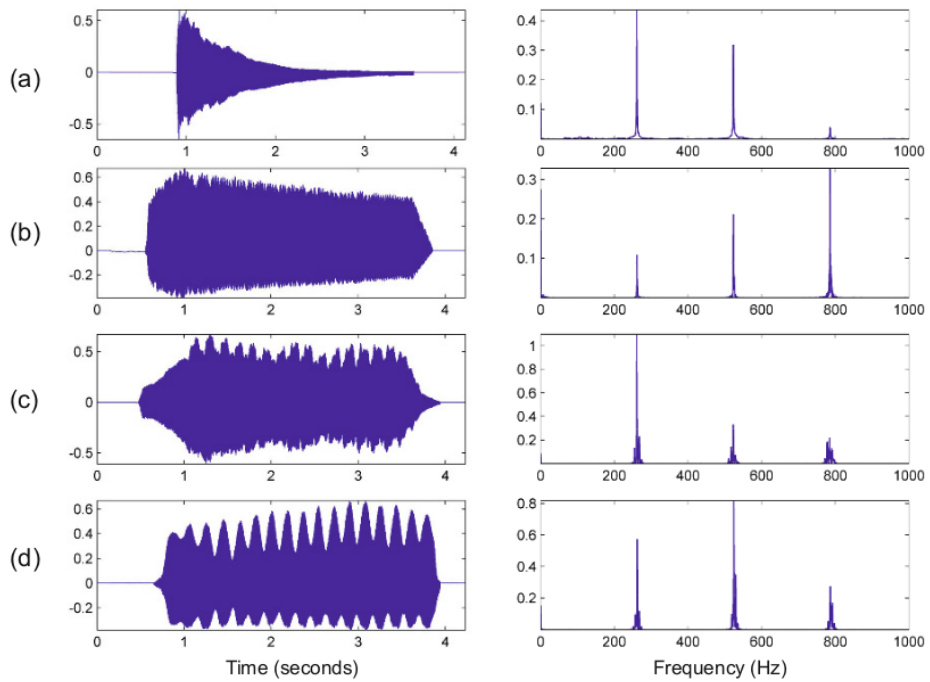



**Fig. 2.5** Waveform and magnitude Fourier transform of a tone C4 (261.6 Hz) played by different instruments (see also Figure 1.23). **(a)** Piano. **(b)** Trumpet. **(c)** Violin. **(d)** Flute.

- The fundamental frequency $\nu_0$ characterizes the pitch of the sound, i.e., the height of the note.
- The shape of the harmonics profile, or spectral envelope (with harmonics located at multiples of $\nu_0$), characterize the timbre of the sound.
- Timbre is an essential (but not unique) information to distinguish between the sounds of a same note played by different instruments.

Why is the spectrum of the violin more "blurred" than the one of the piano?

Image credit: M. Müller, Fundamentals of Music Processing, Springer, 2015.

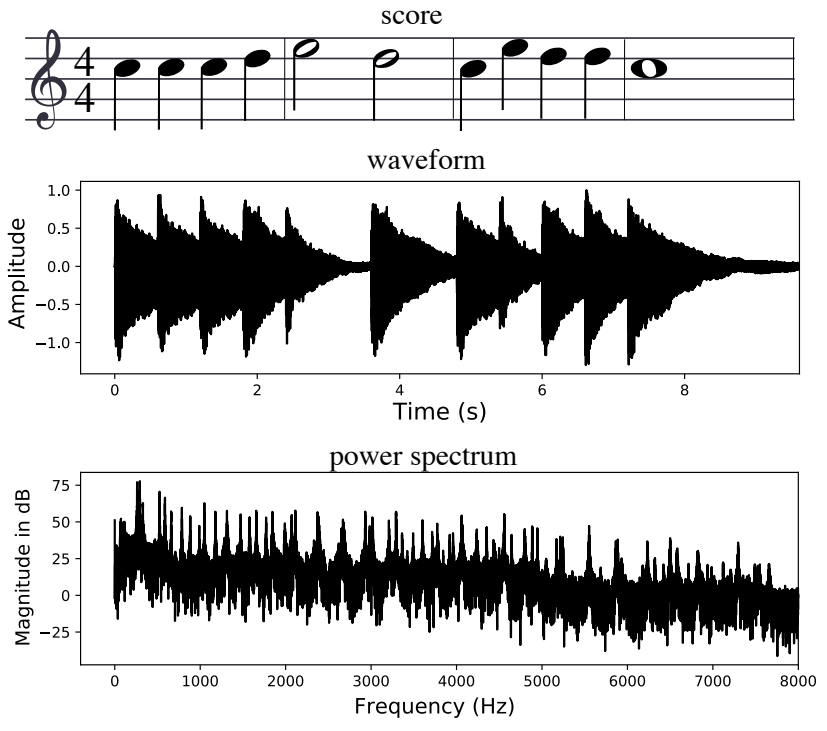
# Short-Time Fourier Transform

J. B. Allen & L. R. Rabiner, A unified approach to short-time Fourier analysis and synthesis, Proceedings of the IEEE, 1977.

E. Jacobsen & R. Lyons, The sliding DFT, Signal Processing Magazine, 2003.

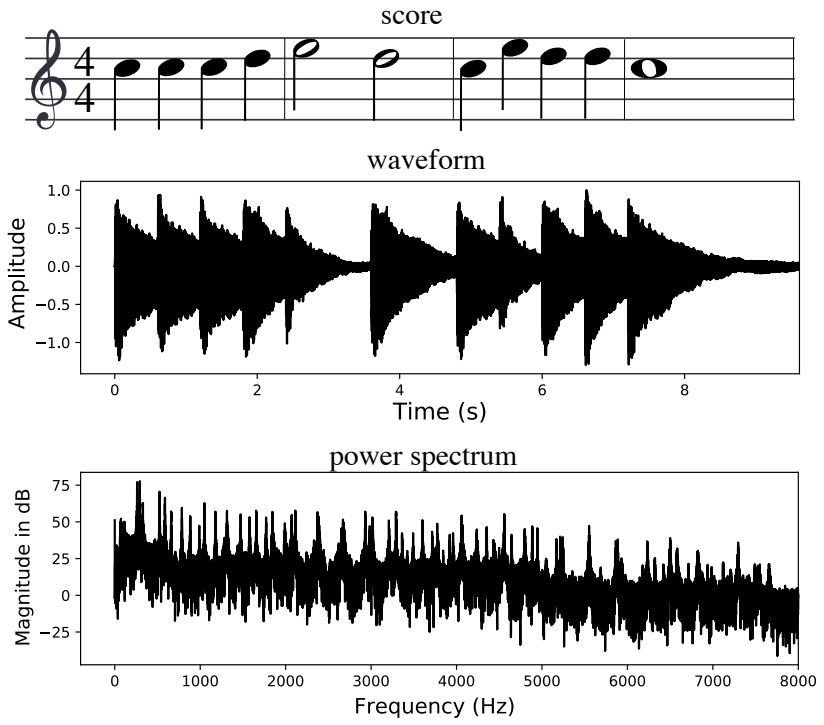
L. Durak & O. Arikan, Short-time Fourier transform: two fundamental properties and an optimal implementation, IEEE Transactions on Signal Processing, 2003.

# Limitations of the DFT




What is missing here?

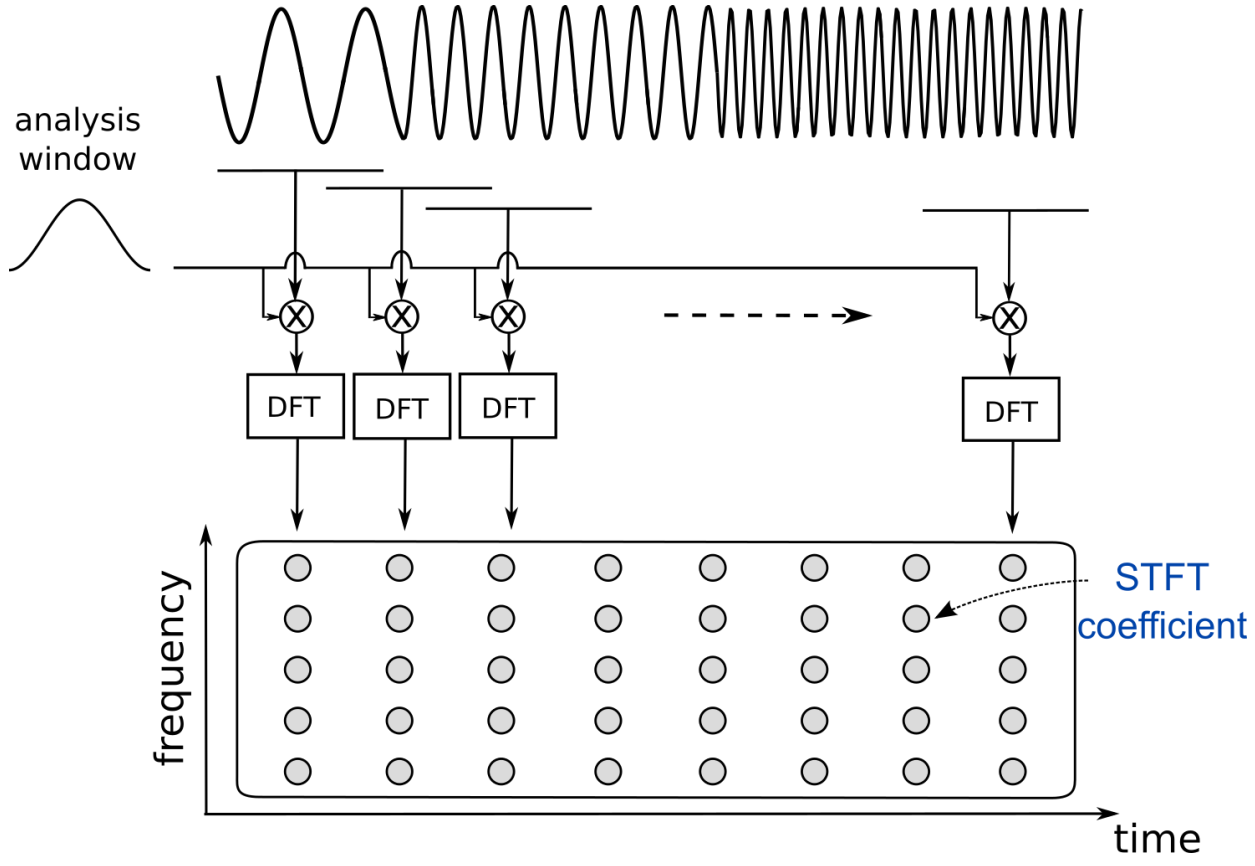
# Limitations of the DFT




- The frequency information is “averaged” over the entire time domain. In other words, the spectral content of the different notes are completely mixed up. This is a mess!
- Local phenomena become global in the DFT when applied that way. We would like to define a “local DFT” and follow its evolution over time.

# Short-Time Fourier Transform (STFT): Principle

The STFT is obtained by computing the DFT on short overlapping smoothed windows of the signal.



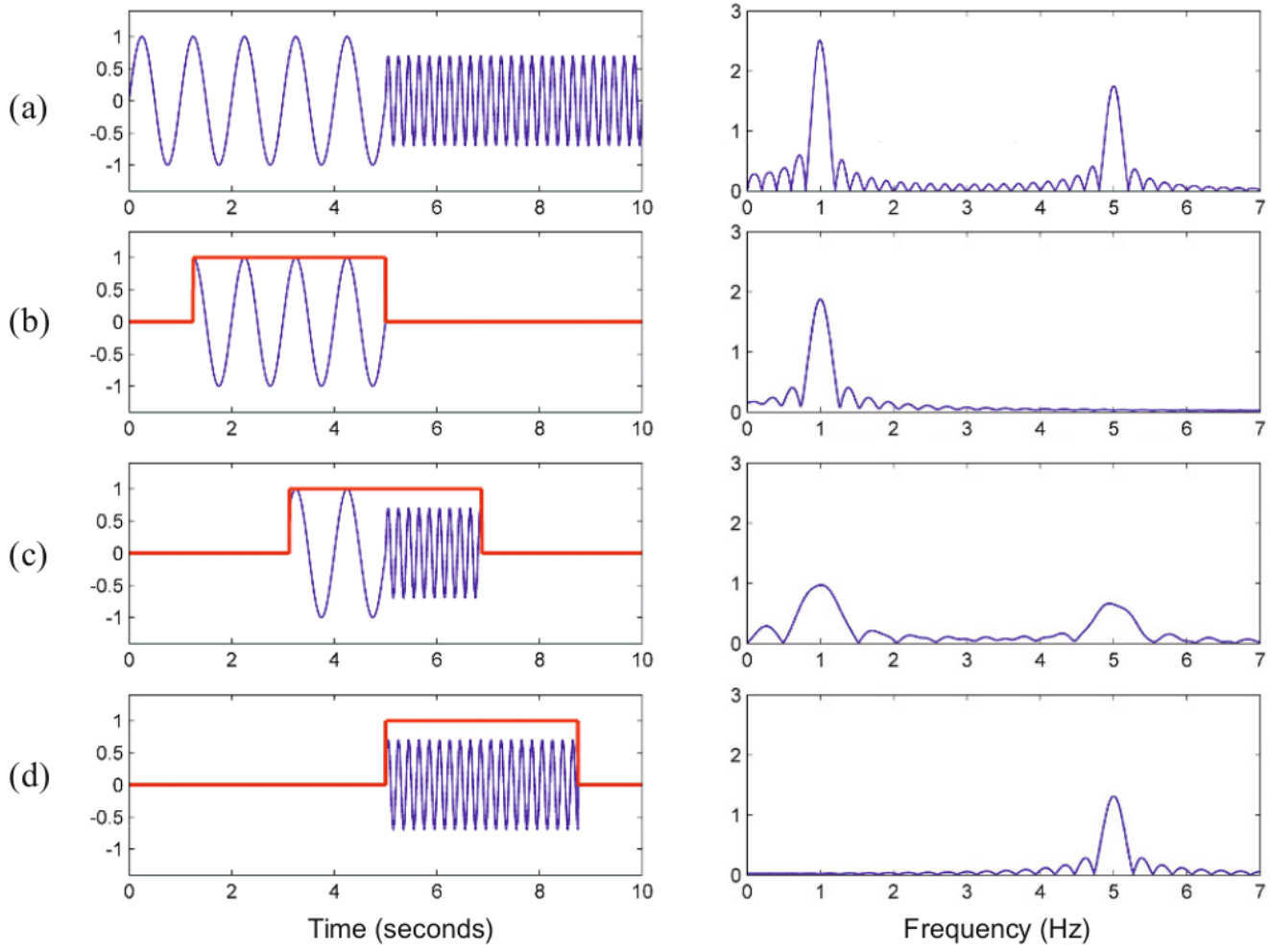



**Fig. 2.8** Signal and Fourier transform consisting of two subsequent sinusoids of frequency 1 Hz and 5 Hz (see Figure 2.6a). **(a)** Original signal. **(b)** Windowed signal centered at $t = 3$. **(c)** Windowed signal centered at $t = 5$. **(d)** Windowed signal centered at $t = 7$.

Image credit: M. Müller, Fundamentals of Music Processing, Springer, 2015.

# STFT: Definition

Let $x(n) \in \mathbb{R}$ be a signal defined in the time domain $n \in \mathbb{Z}$.

An STFT frame is defined for all $m \in \mathbb{Z}$ by:

$$x_m(n) = x(n + mH)w_a(n),$$

- $w_a(n)$ is the analysis window with support $\{0, ..., N-1\}$;
- Therefore, the support of $x_m(n)$ is also $\{0, ..., N-1\}$;
- $H$ is the analysis hop size (increment), with $H < N$, so that there is some overlapp between successive frames, equal to $N - H$ (often expressed in percentage).

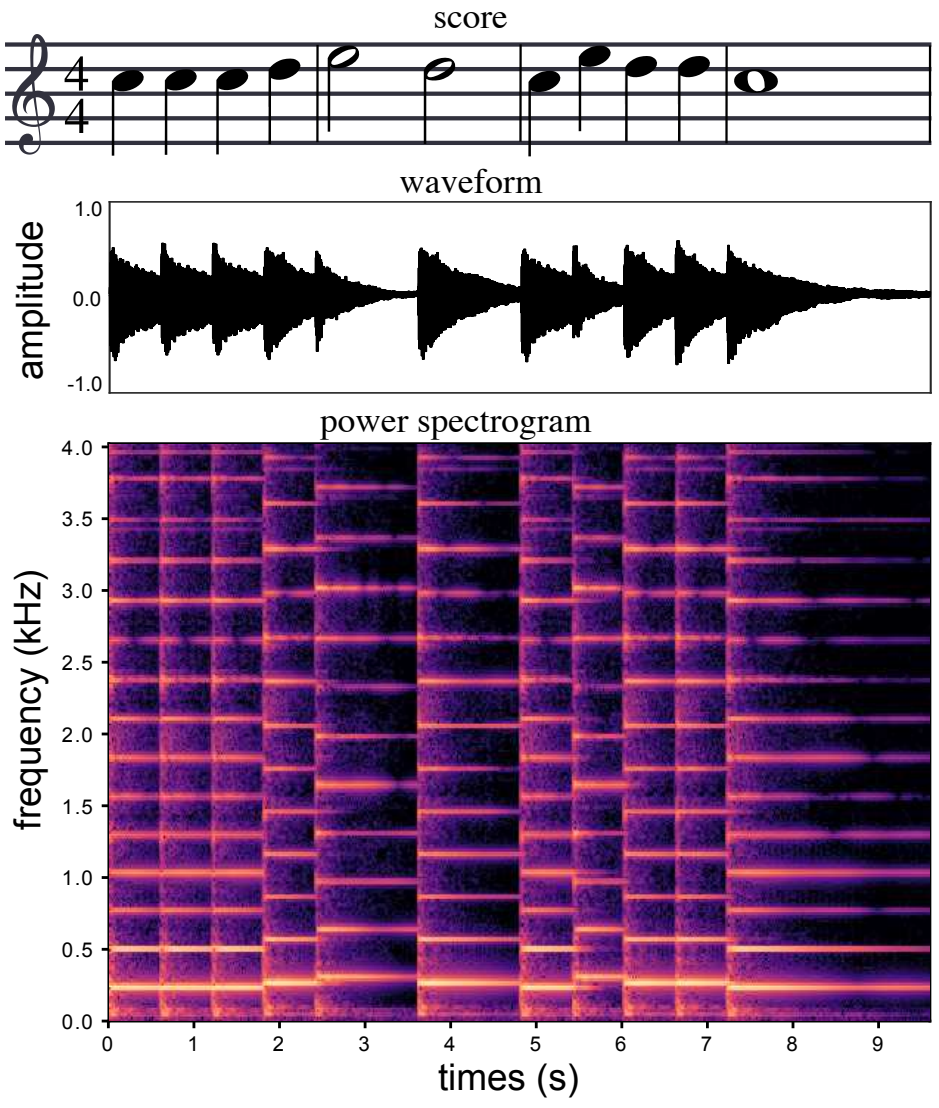
The STFT is defined as the set of DFTs of the frames $x_m(n)$, $m \in \mathbb{Z}$:

$$X(k, m) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_m(n) e^{-j2\pi \frac{kn}{N}}.$$

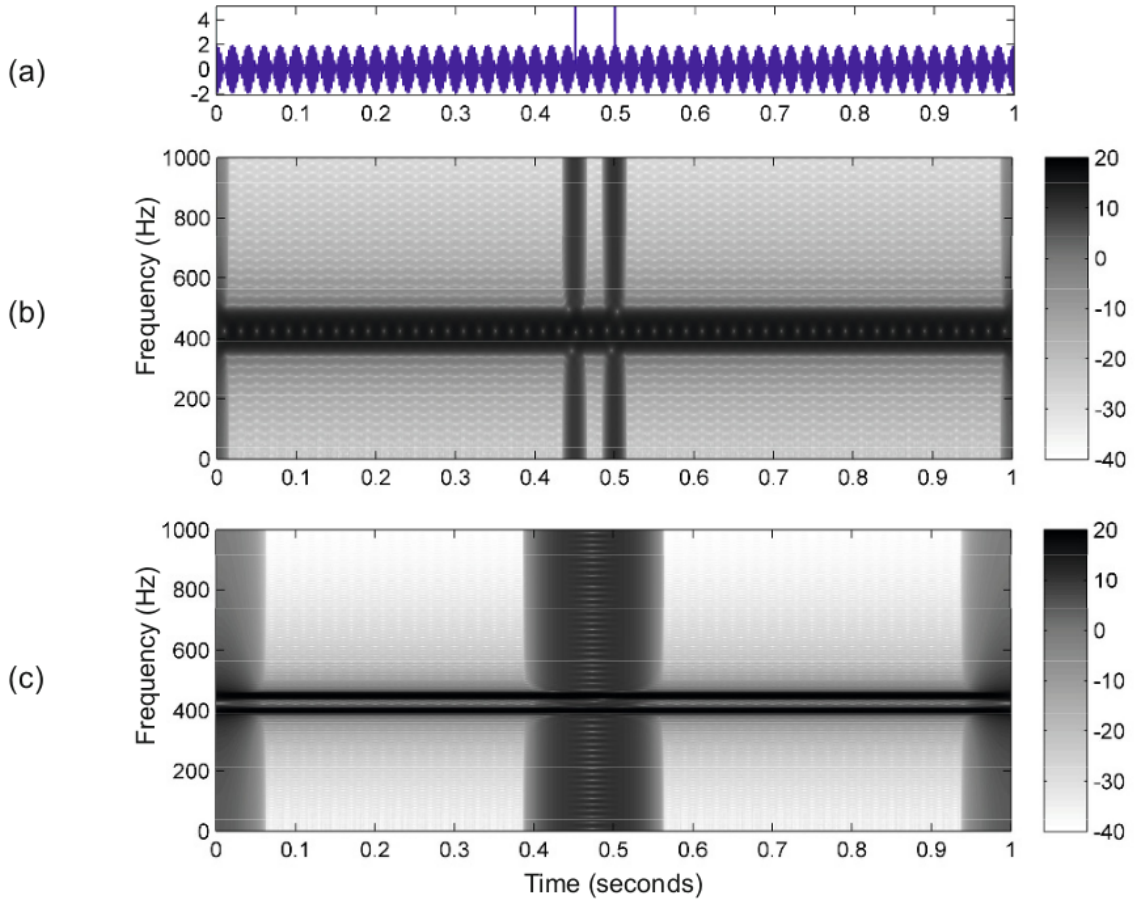
# STFT: Remarks

- We use the same notation $X(\cdot)$ as for the DTFT for simplicity and concision. The fact that we have two "integer" arguments, $k$ and $m$, for the STFT, vs. only one "continuous" argument, $f$, for the DTFT, is enough to differentiate the two transforms.
- In practice, $x(n)$ is of finite length $N_x \gg N$, so that we have a finite, possibly large, number of frames $M = \lceil \frac{N_x}{H} \rceil$ (or something close, depending on, e.g., if we truncate a bit $x(n)$ or complete it with zeros for the last frame).
- The term "STFT" alone designates the complex-valued STFT spectrogram. It is a 2D ($N \times M$) array of complex values. As for the DFT, we differentiate the STFT spectrogram, the STFT magnitude spectrogram, the STFT phase spectrogram, and the STFT power spectrogram.
- In practice, for real-valued signals $x(n)$, we use the Hermitian symmetry to consider only the "lower" part of the STFT spectrogram, i.e. the "positive frequencies", for $k \in [0, \frac{N}{2}]$ ($N$ assumed being even), as we did for the DFT.
- The role of the window function $w_a(n)$ is the same as for the DFT plus it helps ensuring the "continuity" of the successive spectra by smoothing the transition between successive

# The STFT power spectrogram, a "meaningful" representation



# Time-frequency resolution trade-off




- Figure (a): signal made of two sinusoids with close frequencies and two impulses.
- Figures (b) and (c): Spectrograms computed with two different window lengths.
- Short window: good temporal resolution, bad frequency resolution
- Long window: good frequency resolution, bad temporal resolution

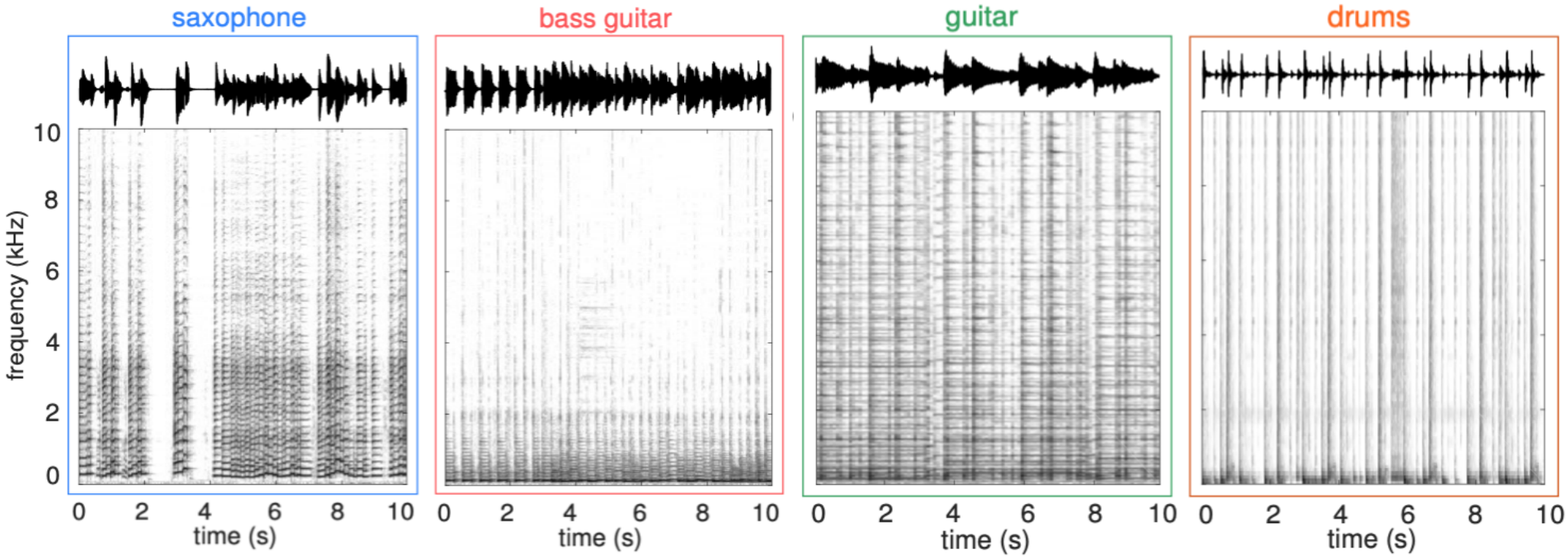
# Typical STFT settings for speech/audio processing

- Speech and most audio signals are non-stationary, with various "speed of variation".
- The analysis window size must be adapted to follow the non-stationarity, i.e., good temporal resolution, while ensuring sufficient frequency resolution.
- For $16$-kHz speech: $N = 512$ ($32$ ms) is the "default" (best) setting.
- For music and general audio, $F_s$ is larger, and the "local stationarity" can be more variable than for speech depending on the audio content (e.g., the average duration of notes in music signals depends on the music style. Adapting the window size to the average note duration is a good idea, but we may want the STFT analysis to capture the note onsets).

  We might have $N = 1024$ or $N = 2048$.
- We often have $H = N/2$ ($50\%$-overlap), though other settings can be used (a smaller $H$ value increases the computation cost).

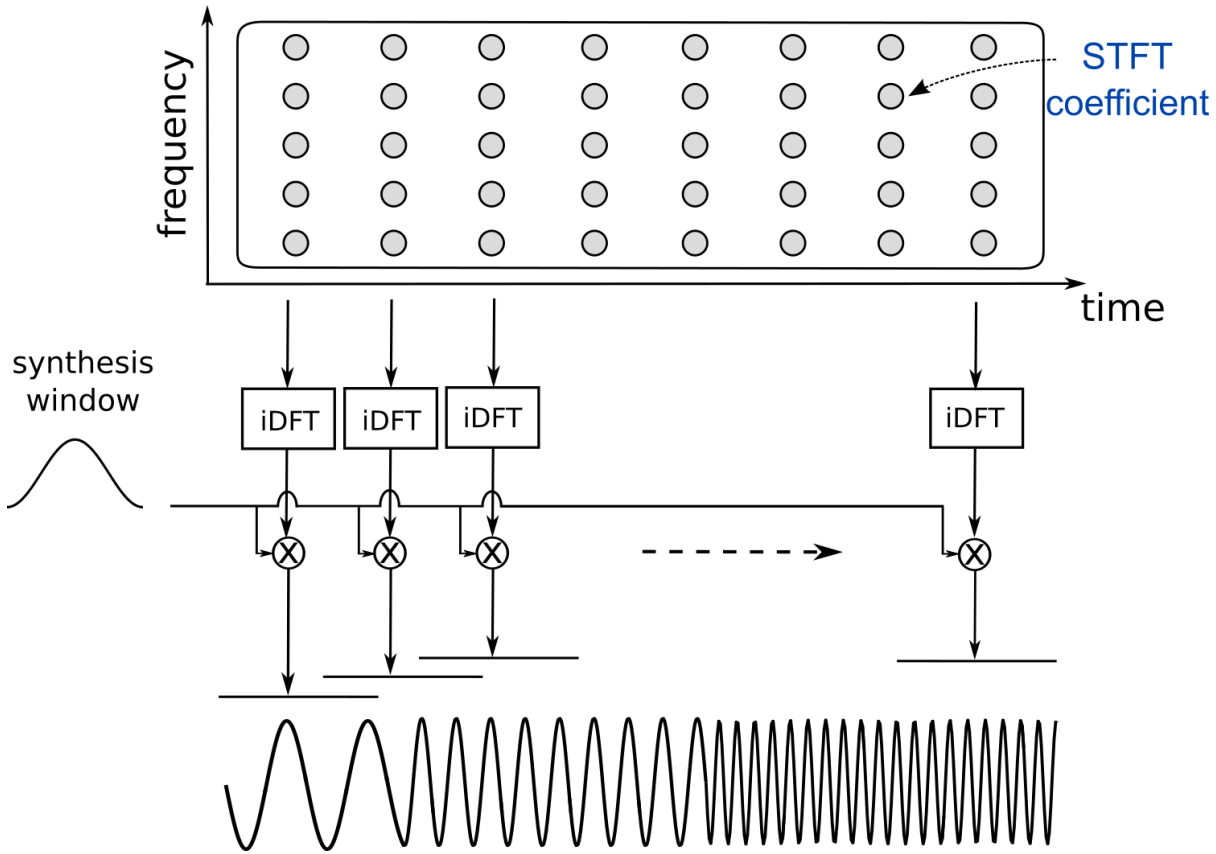
# Example: (pop) music instruments

Much easier to discriminate between different sounds/instruments from the power spectrogram than from the waveform.



# Inverse STFT: Principle

The inverse STFT is computed by taking the inverse DFT of the spectra at all time frame indices and by overlap-add.



# Inverse STFT: Definition

For each frame $m \in \{0, \ldots, M-1\}$ of the STFT, we first compute the inverse DFT:

$$x_m(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X_N(k, m) \exp\left(+j2\pi \frac{kn}{N}\right)$$

The inverse STFT is then computed by overlap-add, for all $n \in \{0, \ldots, N_x\}$:

$$\widehat{x}(n) = \sum_{m=0}^{M-1} w_s(n - mH) x_m(n - mH),$$

where $w_s(n)$ is a smooth synthesis window with the same support as the analysis window $w_a(n)$.

# Perfect reconstruction

We can show that perfect reconstruction is achieved, i.e. $\widehat{x}(n) = x(n),\ \forall n \in \mathbb{Z}$, if

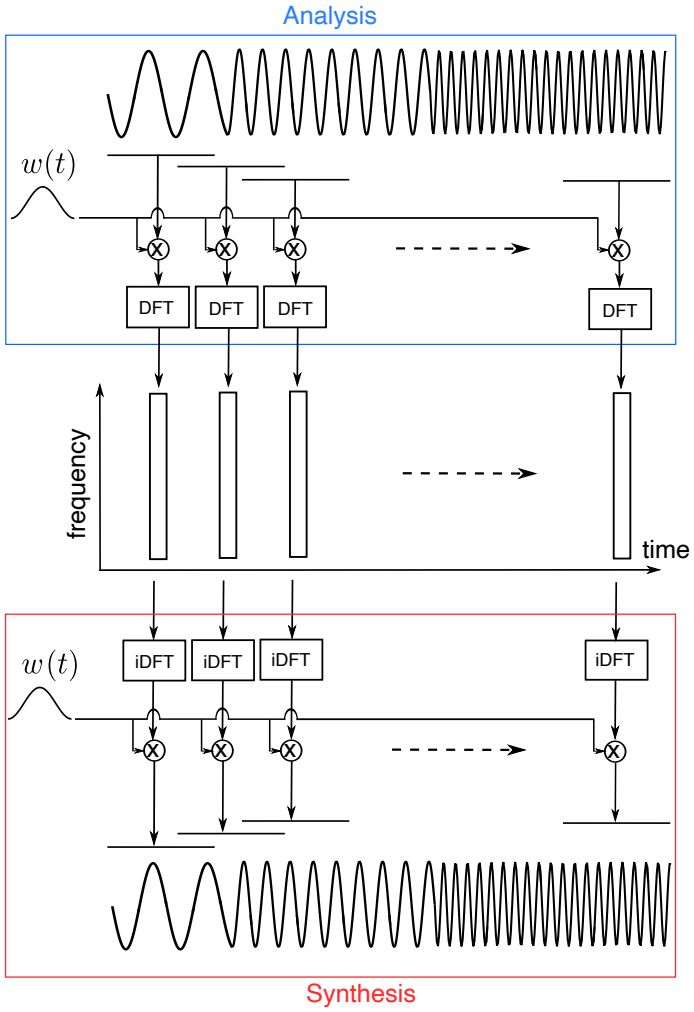
$$\sum_{m\in\mathbb{Z}} w_a(n - mH)w_s(n - mH) = 1, \quad \forall n \in \mathbb{Z}.$$

This is the case for instance with the sine window defined by:

$$w_a(n) = w_s(n) = \begin{cases} \sin\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)\right) & \text{if } 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases},$$

and with $H = N/2$ (i.e., $50\%$-overlap).

# Analysis-transformation-synthesis



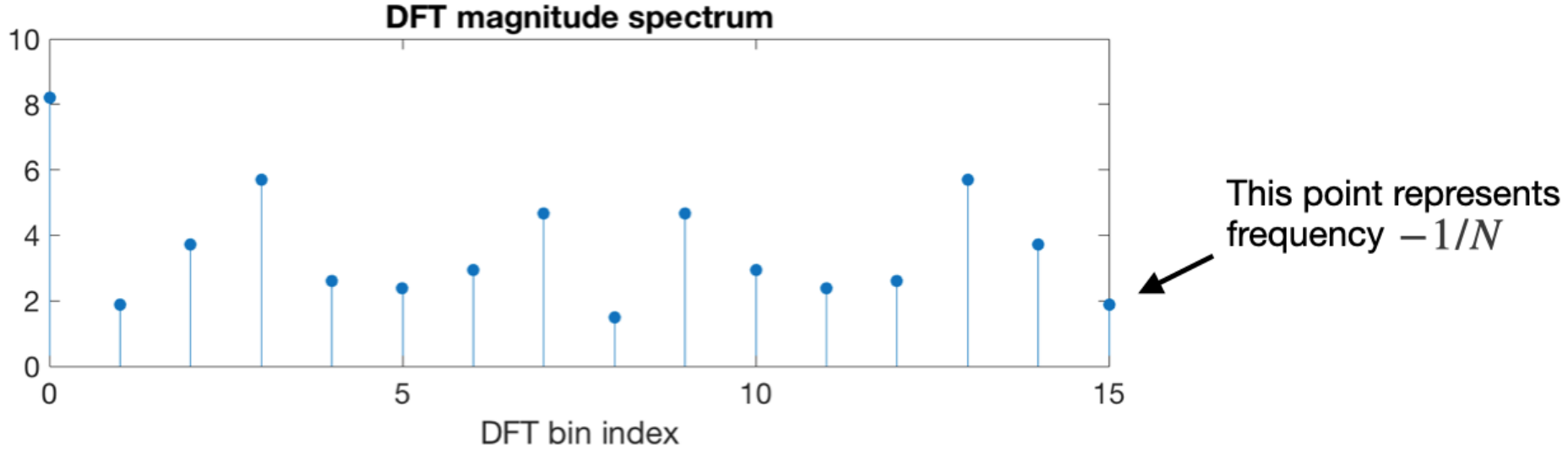

- In many applications, the processing / transformation of speech/audio signals is made in the STFT domain.
- A few examples:
  - The phase vocoder (see next slides)
  - Voice conversion
  - Denoising (e.g., speech enhancement) and dereverberation
  - Source separation (e.g., instrument separation from music mix).
- Note: In case of signal transformation, the synthesis window is very useful to smooth the edges appearing at frame boundaries!

# Analysis-transformation-synthesis

- Short-term analysis, processing, and (re)synthesis is strongly related to real-time processing.
  - This is a manner to implement real-time processing with a joint processing of several signal samples (as opposed to sample-by-sample processing).
  - The "current" block of signal goes into the input buffer, while the previous block is processed, and the even previous processed block goes to the output buffer.
  - Each block is processed (almost) independently of the others, only the overlap-add process involves several consecutive blocks.
  - The process is real-time if the duration of processing one block is lower than the shift between consecutive blocks ($H \times T_s$). The latency is limited to one or two blocks, which is not perceived in speech/audio processing.
- This will be extensively illustrated in the Real-Time Signal Processing course and lab work (STFT-domain filtering).

# Exploiting the STFT symmetry

- Every STFT-domain processing of real-world sounds (real-valued) can be limited to frequency bins in $[0, \frac{N}{2}]$ ($N$ being assumed even).
- The STFT for $k \in [\frac{N}{2} + 1, N - 1]$ can be deduced by symmetry.
- Be careful that the symmetry is "slightly shifted" by bin $0$ (because the DFT is $N$-periodic).




- Be careful that the DFT magnitude is even-symmetric but the STFT phase is odd-symmetric.

# The Phase Vocoder

- Historical example of the use of STFT for the transformation of both speech and music signals
  - Quite "old" and widely-used 100% signal processing technique
  - An extensive literature through the years (many sophisticated improvements have been proposed).
- Two major transformations (and a few minor ones)
  - Time-stretching (change of signal duration) based on interpolation/resampling of STFT frames along the time axis
  - Pitch-shifting (signal transposition) based on rescaling (warping) of STFT frames along the frequency axis

J.L. Flanagan & R.M. Golden, Phase vocoder, Bell System Technical Journal, 1966.

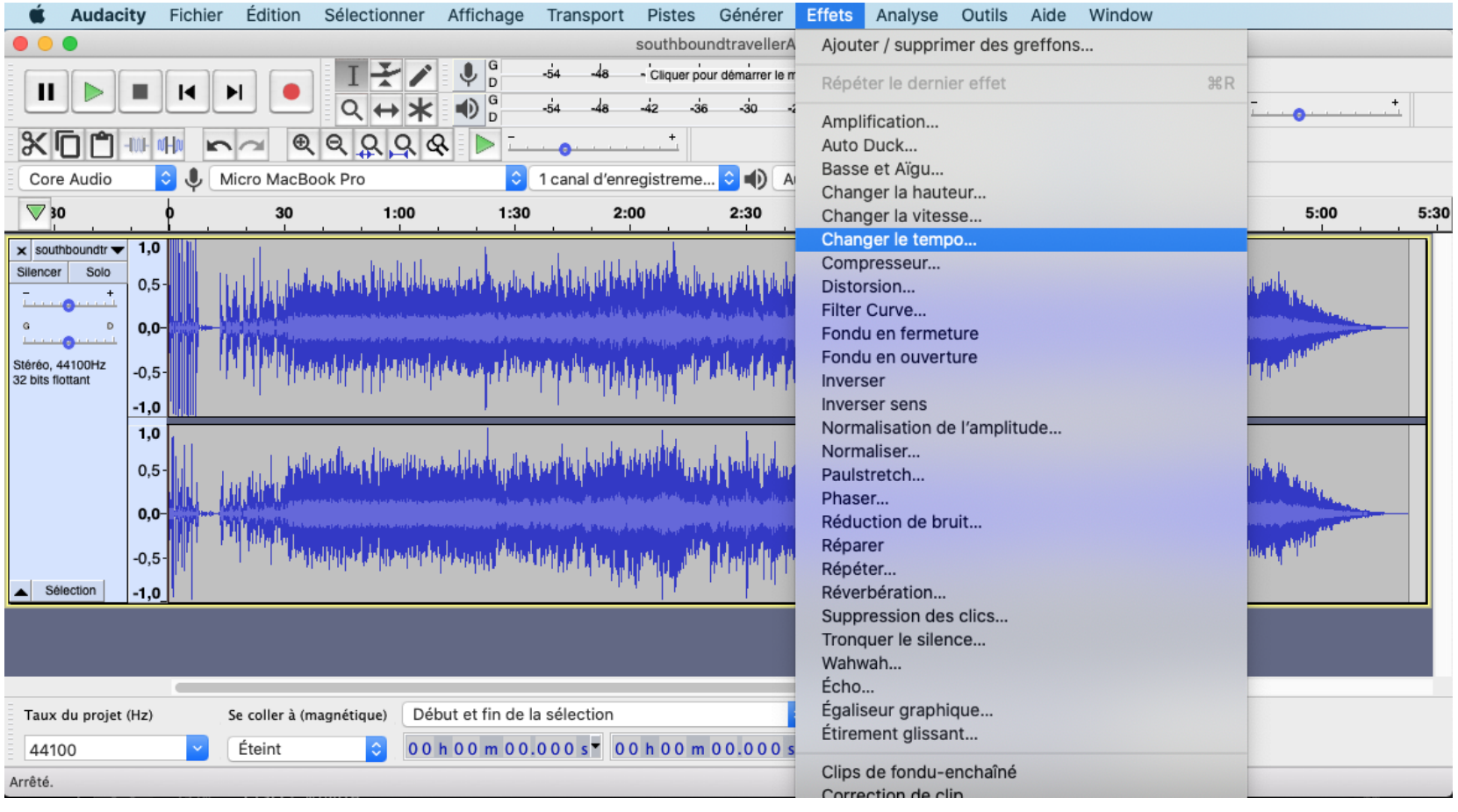
M. Portnoff, Implementation of the digital phase vocoder using the fast Fourier transform, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1976.

J. A. Moorer, The use of the phase vocoder in computer music applications, Audio Engineering Society Convention, 1976.

J. B. Allen, Short time spectral analysis, synthesis, and modification by discrete Fourier transform, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1977.

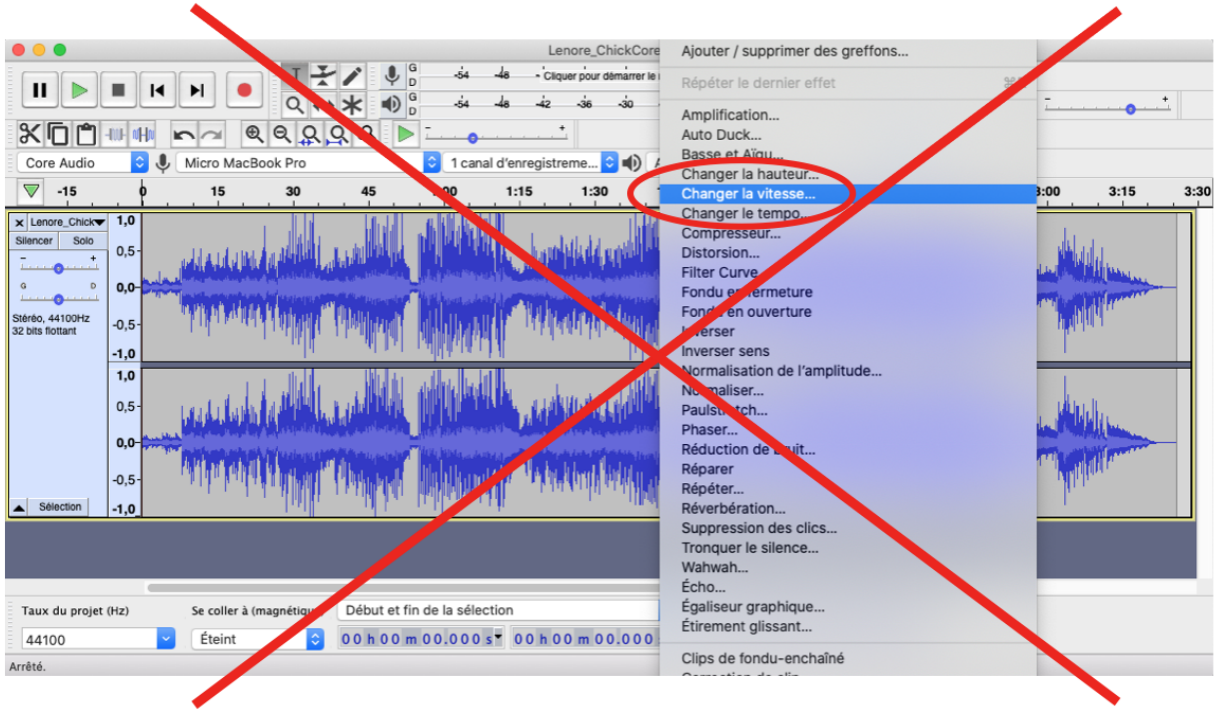
M. Dolson, The phase vocoder: A tutorial, Computer Music Journal, 1986.

# The Phase Vocoder: Audacity demo



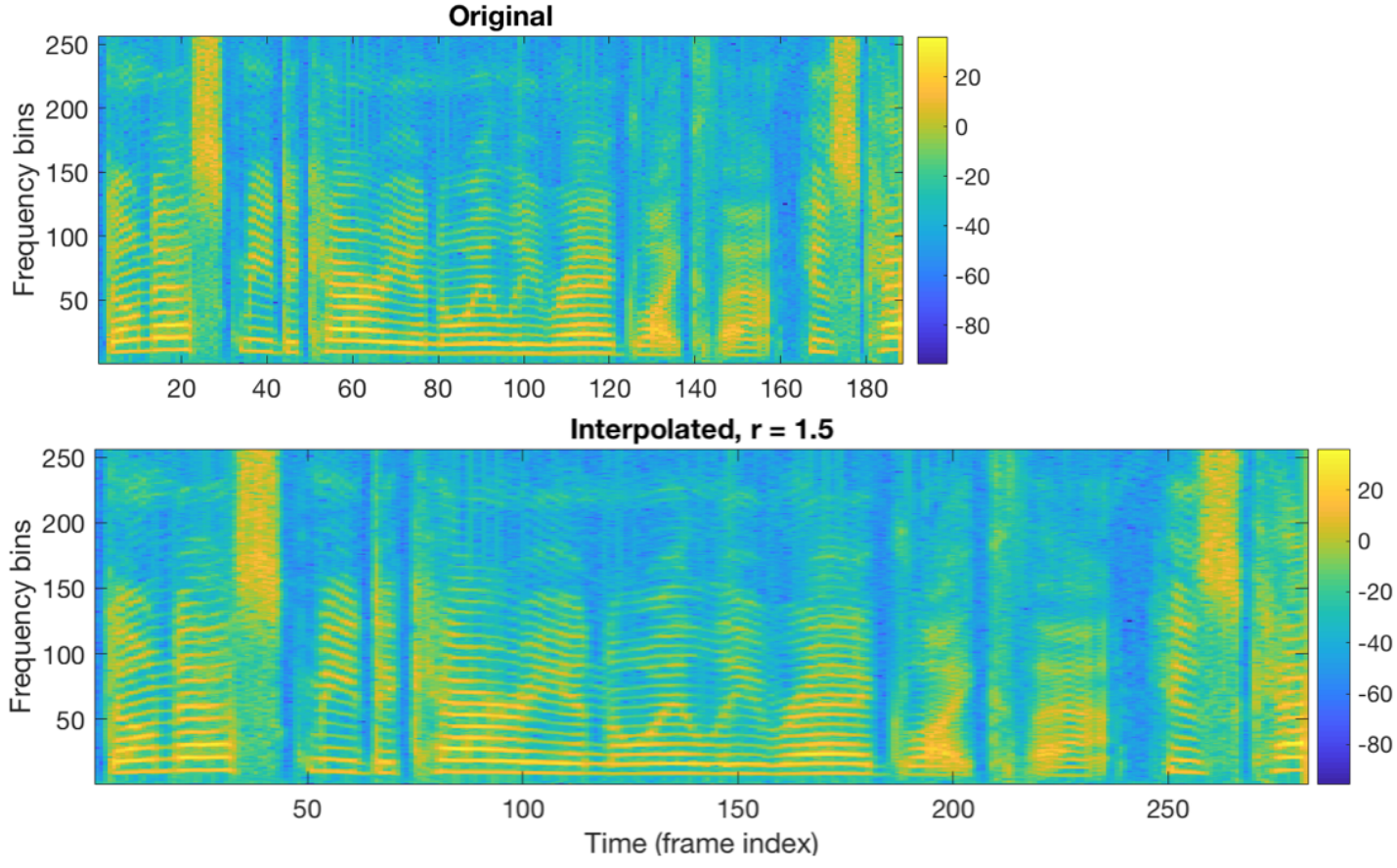
# Before we move forward

Time-stretching or pitch-shifting based on a simple resampling of the signal at a new sampling frequency and playing at the old sampling frequency does **NOT** work (this shifts the whole spectrum along the frequency axis).



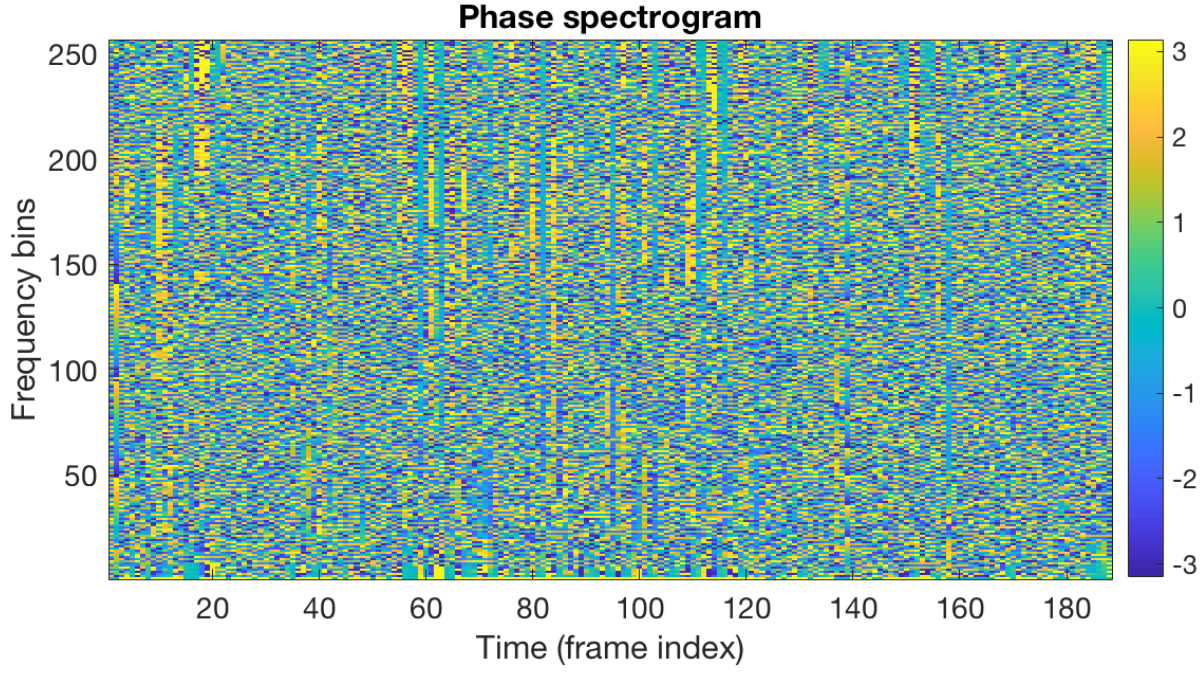
# The Phase Vocoder: Detailing a little bit the time-stretching

- Classical/simple interpolation/resampling of STFT frames along the time axis works well for the magnitude coefficients $|X(k,m)|$, e.g., linear interpolation on a linear scale or on a log scale.



# The Phase Vocoder: Detailing a little bit the time-stretching

However, this does NOT work (at all) for the phase coefficient $\phi(k,m) = \angle X(k,m)$.




- The phase coefficients must be interpolated with great care, taking into account their physical meaning (hence the name of the technique).

# The Phase Vocoder: Phase/frequency relationship

- For a (real-valued) sinusoidal component at frequency $f_k$ in the $k$-th DFT bin, and at frame $m$, we have: $x_{k,m}(n) = 2|X(k,m)| \cos\left(2\pi f_k n + \phi(k,m)\right)$.
- $\phi(k,m)$ is also called the phase at the origin (of the analysis window). From the origin of the current ($m$-th) window to the origin of the next window, the phase grows by $2\pi f_k H$. We thus should have:

$$\phi(k, m+1) = \phi(k,m) + 2\pi f_k H.$$

- In other words, the phase, seen as a function of time, is the summation of frequency over time, with values at frame boundaries that are predictable from the previous phase values and the frequency value.
- Vice-versa, the frequency $f_k$ is predictable from two consecutive phase values:

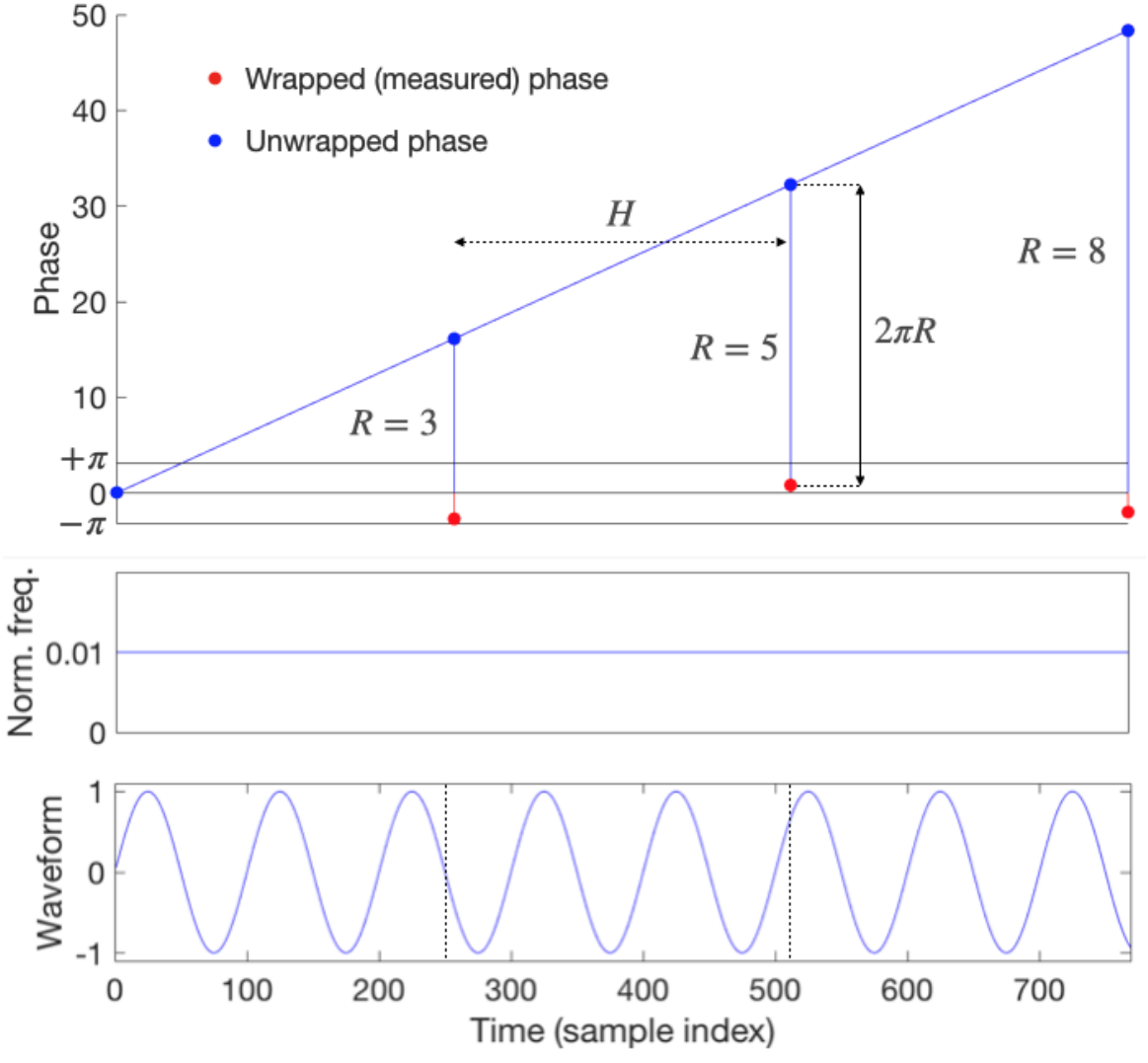
$$f_k = \frac{\phi(k, m+1) - \phi(k,m)}{2\pi H}.$$

# The Phase Vocoder: Wrapped and unwrapped phase

- A major general problem of phase manipulation in audio analysis/transformation/synthesis applications is that the STFT phase coefficients $\phi(k, m)$ are measured independently for each STFT frame and do not take into account the previous frames.
- As a result, the values are provided within $]-\pi, +\pi]$ and they represent the "true phase values" (resulting from summation of frequency over time) up to an integer multiple of $2\pi$.
- In the audio processing context, a phase (measured) value within $]-\pi, +\pi]$ is called the wrapped phase, and the corresponding "true phase value" (resulting from summation of frequency over time) is called the unwrapped phase.
- The unwrapped phase is obtained from the wrapped phase by unwrapping it for an integer multiple of $2\pi$. The relationship between two consecutive (measured) wrapped phase values is thus actually:

$$\phi(k, m+1) + 2\pi R(k, m) = \phi(k, m) + 2\pi f_k H, \quad \text{with} \quad R(k, m) \in \mathbb{Z}^+.$$

# The Phase Vocoder: Wrapped and unwrapped phase




Note: In this figure, the unwrapping process is "cumulated" from the first frame $m = 0$.

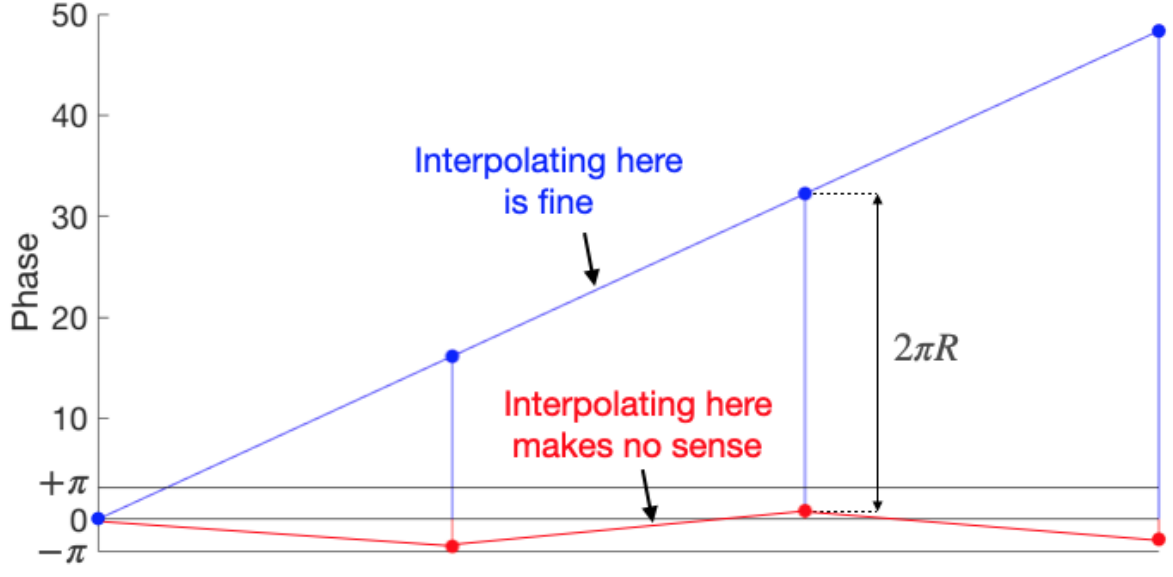
(And so is $R$).

This leads to the complete "physically" meaningful phase trajectory.

# The Phase Vocoder: Interpolating the phase over time

- This explains why we must not directly interpolate STFT phase coefficients (i.e., wrapped phase values) over time: This generates values that are not sampled on a consistent phase trajectory over time, and that are almost equivalent to random.

> The physically meaningful interpolatable values are the unwrapped phase values.



# The Phase Vocoder: Interpolating the phase over time

- Remaining problem: We do not know $R(k, m)$ and $f_k$.
- The good news: We can easily derive a method to jointly estimate them from the measured STFT phase values using the phase/frequency general relationship.
- Note: We will actually need to know $f_k$ to process the signal time-streching.

# The Phase Vocoder: Estimation of $R(k,m)$ and $f_k$

- From the previous slides, we have: $\phi(k,m+1) - \phi(k,m) + 2\pi R(k,m) = 2\pi f_k H.$
- We easily deduce: $\underbrace{\phi(k,m+1) - \phi(k,m) - 2\pi \frac{k}{N} H}_{d\phi_r(k,m) = \text{the phase remainder}} + 2\pi R(k,m) = 2\pi \big(f_k - \frac{k}{N}\big) H.$
- The width of a frequency bin is $\frac{1}{N}$. Therefore $f_k - \frac{k}{N} \in [-\frac{1}{2N}, \frac{1}{2N}]$. If we assume $H \leq \frac{N}{2}$, we have $2\pi \big(f_k - \frac{k}{N}\big) H \in [-\pi, \pi]$.
- To find $R(k,m)$, we thus just have to count how many times we have to add $2\pi$ to $d\phi_r(k,m)$ to make it go into $[-\pi, \pi]$. This is a wrapping operation. In short:

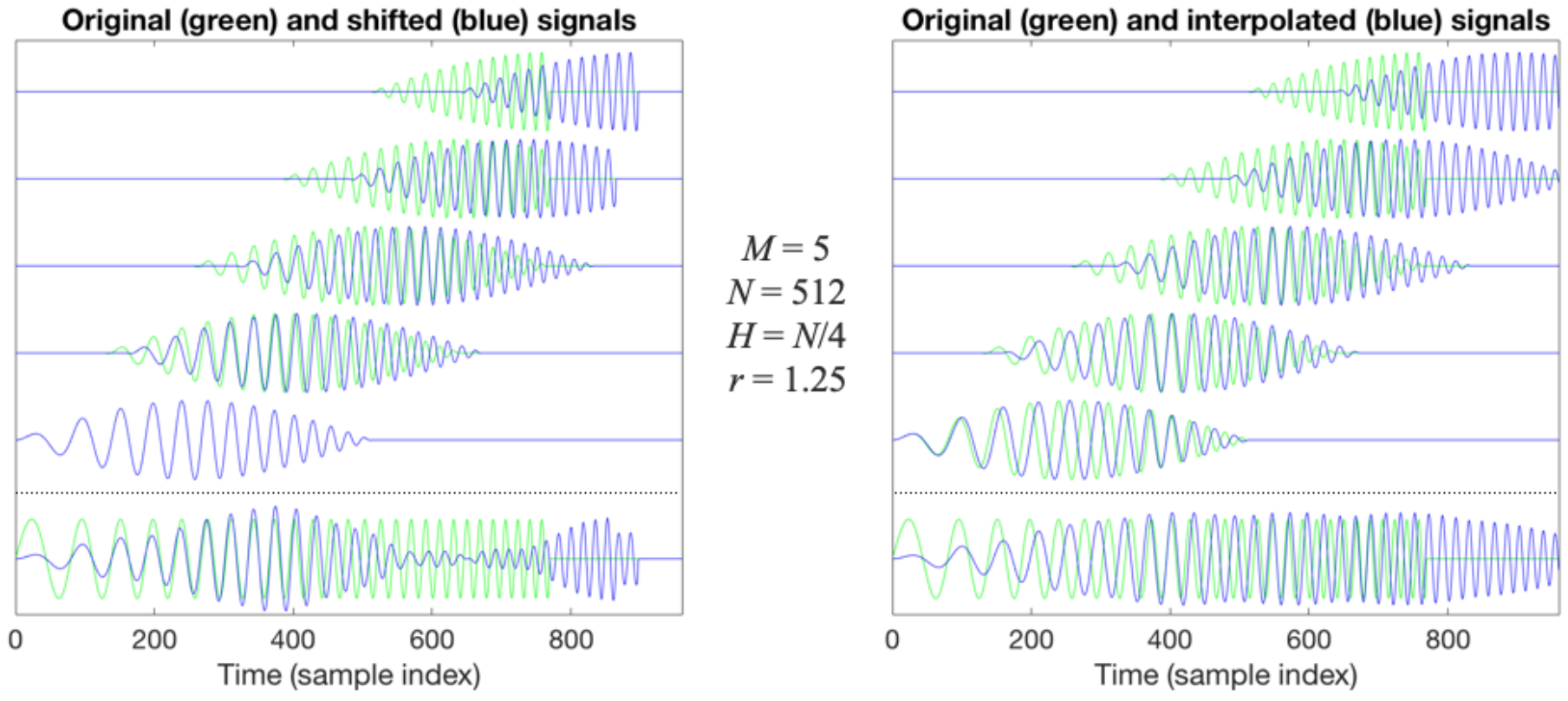
$$2\pi \big(f_k - \frac{k}{N}\big) H = \text{wrap}\left[d\phi_r(k,m)\right] \qquad R(k,m) = \frac{\text{wrap}\left[\, d\phi_r(k,m)\right] - d\phi_r(k,m)}{2\pi}.$$

- We deduce the frequency estimate: $\hat{f}_k = \frac{k}{N} + \frac{\text{wrap}[\, d\phi_r(k,m)]}{2\pi H}.$

# The Phase Vocoder: Phase interpolation over time and signal time-stretching

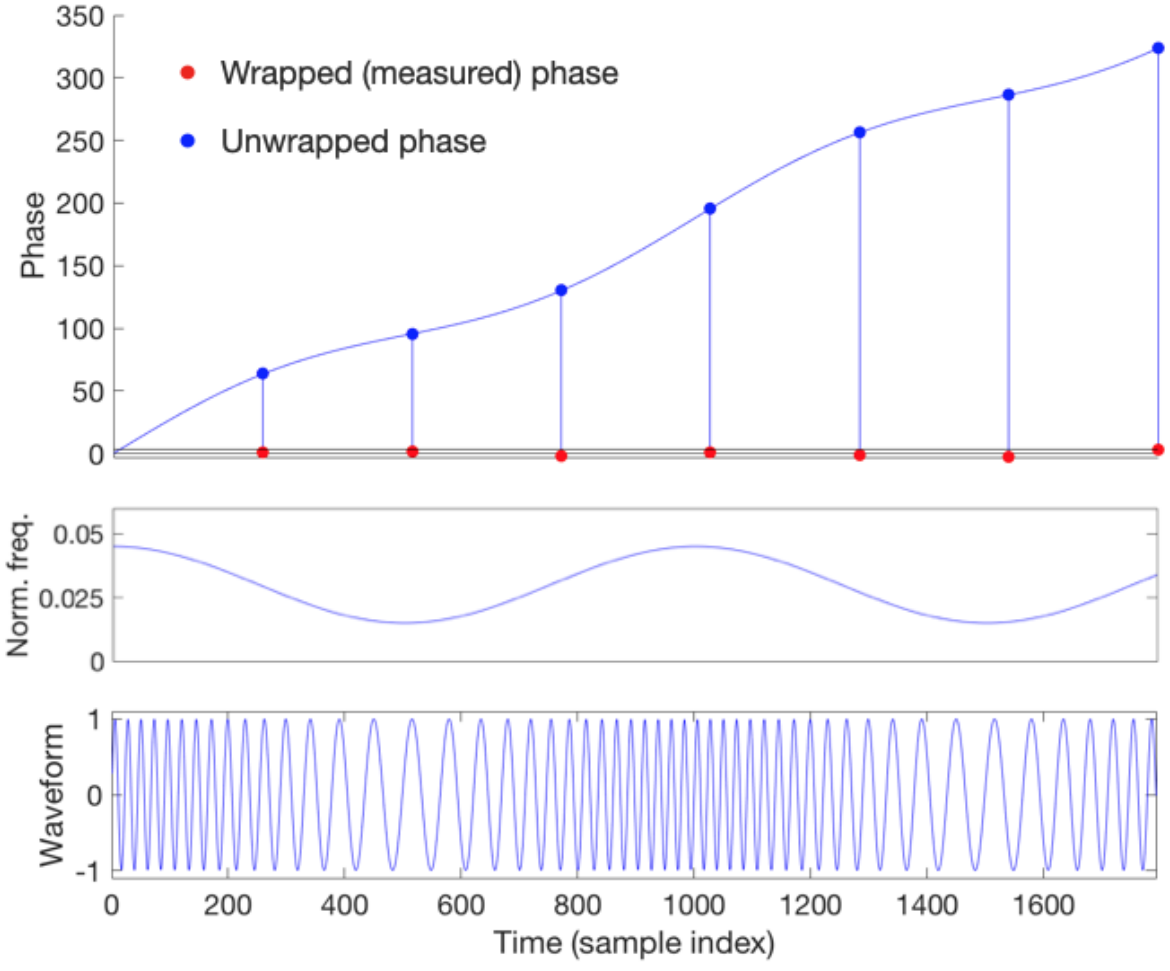
- Now, we can compute the phase $\phi_i(k, m, N_i(m))$ at any arbitrary new time instant $N_i(m)$ after the origin of the current $m$-th frame: $\phi_i(k, m, N_i(m)) = \phi(k, m) + 2\pi \hat{f}_k N_i(m)$.
- If we want to modify the length of the signal by a fixed factor $\frac{H_i}{H}$:
  - Start from the first frame $m = 0$
  - For all bins $k$, compute $\hat{f}_k$ with the method from previous slide and compute the phase after $H_i$ samples: $\phi_i(k, 1, H_i) = \phi(k, 0) + 2\pi \hat{f}_k H_i$.
  - Reiterate on the next frames using $\phi_i(k, m-1, H_i)$ as the starting phase.
  - Finally, inverse STFT is applied on the interpolated phase (and original magnitude) coefficients, with the new hop size $H_i$.
- If $H_i < H$, the signal is shortened. If $H_i > H$, it is extended. (And if $H_i = H$, it is left unchanged :-).

# The Phase Vocoder: Illustration of time-stretching




Note: This is a simulation made with a very "amateur" phase vocoder implemented in Matlab in a couple of hours.

# The Phase Vocoder: Generalization to time-varying frequency




- So far, we have (more or less implicitely) considered that the frequency $f_k$ is constant over time.
- The same general framework is valid for a ("continuously") time-varying frequency:

$$x(n) = A\cos\left(2\pi \sum_{l=0}^{n} f(n) + \phi_0\right)$$

- Here, $f(n)$ is the instantaneous frequency, and $\hat{f}_k$ represents the average frequency of the component in bin $k$.

# A remark about phase processing in the deep learning era

- After 50 years of signal processing, phase is still poorly considered in audio processing.
- This can be explained by two main reasons:
  - Very rapidly evolving parameter, more difficult to capture and process than magnitude (e.g., in denoising)
  - The "unimportance" of phase in the perception of speech/audio signals; be careful there! This is often a misconception.
- This is still true at the deep learning era: The processing of phase with DNNs is still a quite open topic of the scientific literature.
- Understanding the phase vocoder (a quite old school signal processing tool) means understanding what the phase really is in audio signals.

# Another example (as a nice transition toward the next section)

Time-frequency masking: Exploit the individual time-frequency characteristics of the source signals to build a mask that is applied to the mixture in the STFT domain.

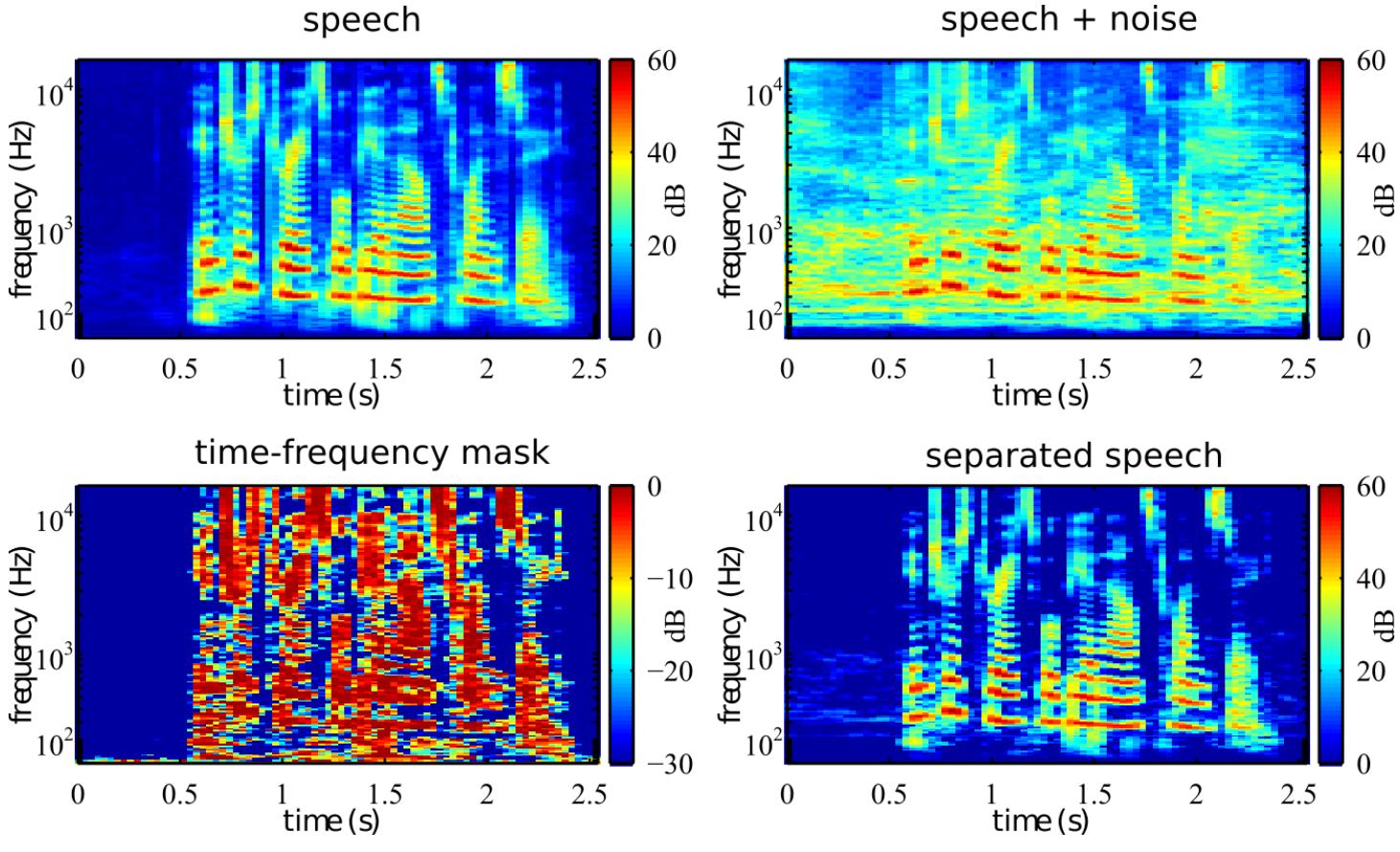



Image credit: J. Le Roux, E. Vincent & H. Erdogan, Learning-based approaches to speech enhancement and separation, Tutorial at Interspeech Conference 2016.

# Part 3: Speech/audio denoising and separation

(single-channel configuration)

# Definitions

- Denoising: Estimation of a clean signal from a noisy observation of it.
- In audio processing, the noise is generally some "background" or "environmental" noise, whereas the signal to be cleaned is emanating from a specific source of interest.
- Speech enhancement in noise is a very important particular case, that we will consider in details.
- Source separation rather refers to the separation of signals of the same kind, e.g., different overlaped speech signals in a conversation, or different music instruments in a music mix (all of them can be considered as a signal of interest, and not a "parasite" noise).

J. S. Lim & A. V. Oppenheim, Enhancement and bandwidth compression of noisy speech, Proceedings of the IEEE, 1979.

J. Benesty, S. Makino & J. Chen, Speech enhancement, Springer, 2006.

S. Makino, T. W. Lee & H. Sawada (Eds.), Blind speech separation, Springer, 2007.

P. C. Loizou, Speech enhancement: Theory and practice, CRC Press, 2013.

S. Gannot et al., A consolidated perspective on multimicrophone speech enhancement and source separation, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2017.

E. Vincent, T. Virtanen & S. Gannot (Eds.), Audio source separation and speech enhancement, John Wiley & Sons, 2018.

# Dichotomy of methods

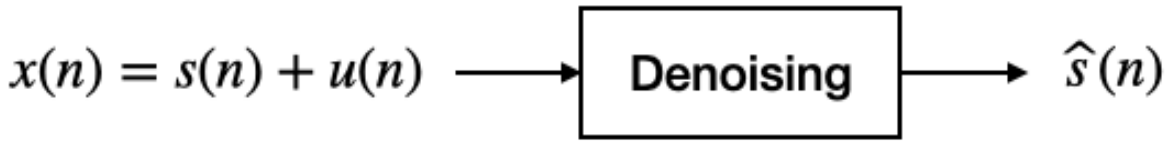
- For both enhancement and separation, there are $2 \times 2$ families of techniques:
  - Single-channel vs. multichannel (with multimicrophone recordings) methods.

    Basically, single-channel techniques exploit the spectral content/structure of signals and multichannel techniques exploit the spatial information across channels (alone or possibly in addition to the spectral information).
  - "Classical" techniques based on signal/channel models (signal processing) and "modern" techniques based on data and deep learning (deep neural networks).

    There exist connections between them.

We will see examples of these different kinds of methods. In this part, we will see single-channel techniques, whereas multichannel processing will be considered in Part 4.

# Speech enhancement in noise



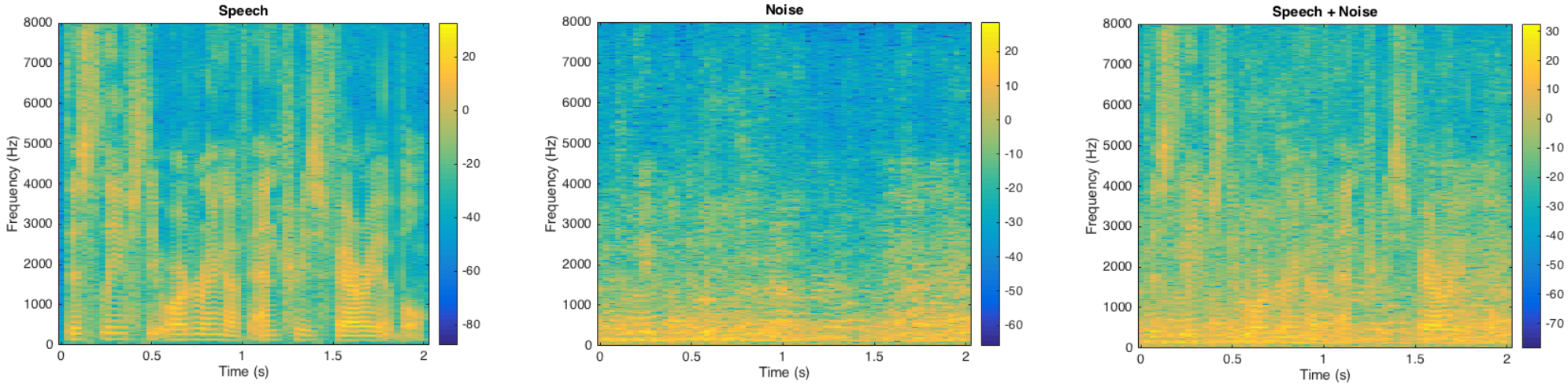

- Goal: Improve the quality and intelligibility of speech signals for Human-to-Human communication (telecommunication systems) and Human-machine communication (automatic speech recognition systems) in noisy environments.
- A very old topic of signal processing. 50 years of scientific/technological literature. Current renewal with "modern" applications (e.g., smart loudspeakers, home assistant) and deep learning.

# Speech enhancement in noise: Classical methods

- 3 historical/classical methods
  - Spectral subtraction
  - Wiener filtering
  - Short-term amplitude estimator
- All 3 methods work in the STFT domain.

  Basically, all methods amount to an estimation problem in the STFT domain.

# Speech enhancement in noise: Hypotheses / principles




- The speech signal is not always active. We can use a Voice Activity Detector (VAD) to detect if the speech signal is present in a frame (or a subset of consecutive frames) or if there is only noise. A VAD is a frame binary classifier (there are plenty of methods).

- Both speech and noise signals are considered as random signals. Speech signals are non-stationary but are assumed locally stationary (on each analysis frame). Speech and noise are assumed uncorrelated. The noise signal is assumed more stationary than the speech signal, i.e., its spectral characteristics are varying more slowly than the speech characteristics (and they are different from speech characteristics).

# Spectral subtraction

- Let us denote by $v_s(k,m) = \mathbb{E}\left[|S_N(k,m)|^2\right]$, $v_u(k,m) = \mathbb{E}\left[|U_N(k,m)|^2\right]$, and $v_x(k,m) = \mathbb{E}\left[|X_N(k,m)|^2\right]$, the power spectral density (PSD) of the speech signal, the noise, and the noisy observation, respectively.
- Since speech and noise are assumed uncorrelated, $v_x(k,m) = v_s(k,m) + v_u(k,m)$.
- An estimate of $v_x(k,m)$ is directly given by: $\widehat{v}_x(k,m) = |X_N(k,m)|^2$.
- An estimate $\widehat{v}_u(k,m)$ of $v_u(k,m)$ is computed during speech inactivity, based on its "slow" variations (details next slide).
- An estimate of $v_s(k,m)$ is thus given by: $\widehat{v}_s(k,m) = \max\left(0, \widehat{v}_x(k,m) - \widehat{v}_u(k,m)\right)$.
- Then we can deduce a clean speech signal estimate: $\widehat{S}_N(k,m) = |\widehat{S}_N(k,m)|e^{j\angle X_N(k,m)}$ with $|\widehat{S}_N(k,m)| = \sqrt{\widehat{v}_s(k,m)}$
- Note: There is no attempt to denoise the phase.

J. S. Lim & A. V. Oppenheim, Enhancement and bandwidth compression of noisy speech, Proceedings of the IEEE, 1979.

# Noise PSD estimate

- An estimate $\widehat{v}_u(k,m)$ is computed during speech inactivity.
- This can be done, e.g., by averaging the short-term (frame-wise) PSD estimates using the most "recent" noise-only frames:

$$\widehat{v}_u(k,m) = \frac{1}{m2 - m1 + 1} \sum_{p=m1}^{m2} |X_N(k,p)|^2,$$

where $\{m1, ..., m2\}$ is the set of indexes of the most "recent" noise-only frames, as provided by the VAD (this estimate is regularly updated).

# Wiener filtering

- The Wiener filter is the minimum mean squared error (MMSE) linear estimate of $s$ given $x$:

$$\widehat{w} = \arg\min_{w} \mathbb{E}\left[|\hat{s}(n) - s(n)|^2\right] \text{ subject to } \hat{s}(n) = w(n) * x(n).$$

- For stationary signals, a bit of maths give the following solution:

$$W(k) = \frac{v_s(k)}{v_s(k) + v_u(k)}.$$

- In practice, the Wiener filter is applied frame-wise, using the local estimates $\widehat{v}_u(k, m)$ and $\widehat{v}_s(k, m)$ provided by the VAD and the spectral subtraction. We have:

$$\widehat{S}_N(k, m) = \frac{\widehat{v}_s(k, m)}{\widehat{v}_s(k, m) + \widehat{v}_u(k, m)} X_N(k, m) = W(k, m) X_N(k, m).$$

- $W(k, m) \in \mathbb{R}^+$, i.e., $w(n)$ is a zero-phase filter, and, again, the phase of the speech signal estimate is equal to the phase of the noisy signal.

J. S. Lim & A. V. Oppenheim, Enhancement and bandwidth compression of noisy speech, Proceedings of the IEEE, 1979.

A. Papoulis, Signal Analysis, McGraw-Hill, 1977.

# Short-Term Amplitude estimator

- Bayesian estimation of $|S_N(k,m)|$: Given a (complex Gaussian) source model $p(S_N(k,m))$ of parameter $\widehat{v}_s(k,m)$, and a (complex Gaussian) noise model $p(U_N(k,m))$ of parameter $\widehat{v}_u(k,m)$, we look for the posterior distribution $p(|S_N(k,m)| \mid X_N(k,m))$ and the corresponding MMSE estimator of $|S_N(k,m)|$:

$$|\widehat{S}_N(k,m)| = \mathbb{E}\big[|S_N(k,m)| \mid X_N(k,m)\big].$$

- Then, again: $\widehat{S}_N(k,m) = |\widehat{S}_N(k,m)|e^{j\angle X_N(k,m)}$.
- Not detailed here. (We will come back to the complex Gaussian distribution later).
- Note: If we look instead for $p(S_N(k,m) \mid X_N(k,m))$, we obtain $\widehat{S}_N(k,m) = \mathbb{E}\big[S_N(k,m) \mid X_N(k,m)\big]$, which corresponds to... the output of the Wiener filter!

Y. Ephraim & D. Malah, Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984.

# Speech enhancement: Performance measures

- Input SNR = ratio of signal power and noise power in the mixture signal:

$$\mathrm{SNR}_{\mathrm{in}} = 10 \log_{10} \frac{P_s}{P_u} \quad \text{with} \quad P_s = \frac{1}{N_x} \sum_{n=0}^{N_x-1} s(n)^2$$

- Output SDR = ratio of signal power and residual noise (= distortion) power in the enhanced signal:

$$\mathrm{SDR}_{\mathrm{out}} = 10 \log_{10} \frac{P_s}{P_e} \quad \text{with} \quad e(n) = \hat{s}(n) - s(n)$$

  - This is an oracle measure ($s(n)$ must be available), used in simulations.
  - Careful: it is not robust to scaling, time shift or dephasing → may need compensation !

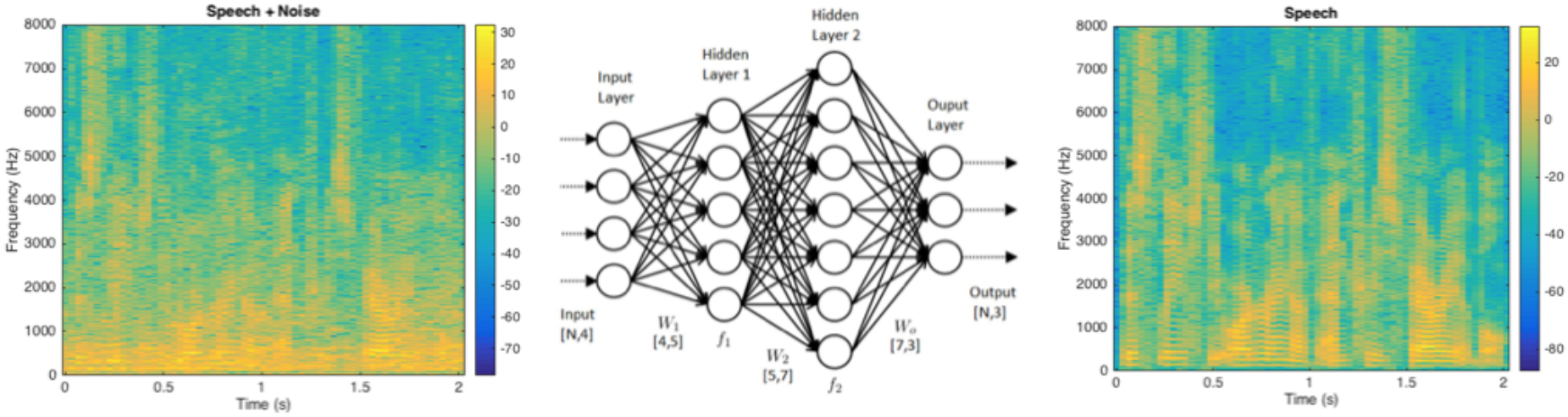
- Gain = $\mathrm{SDR}_{\mathrm{out}} - \mathrm{SNR}_{\mathrm{in}}$. Takes into account the difficulty of the denoising task.
- Can be calculated in the time domain or in the STFT domain, thanks to the DFT "orthogonality" and Persival. Averaged frame-wise versions are called segmental SNR/SDR.

# Speech enhancement: The Deep Learning approach

- General principle: Speech enhancement is turned into a supervised data-driven regression problem using deep neural networks (DNNs).
- This is what we do in the deep-learning-based speech/audio processing project (projet de simulation logicielle).
- Two strategies:
  - Direct noisy-2-clean regression from noisy speech power/magnitude spectrogram to clean speech power/magnitude spectrogram + use of the phase of noisy signal
  - Regression from noisy speech power/magnitude spectrogram to a time-frequency mask and application of the estimated mask to the noisy signal

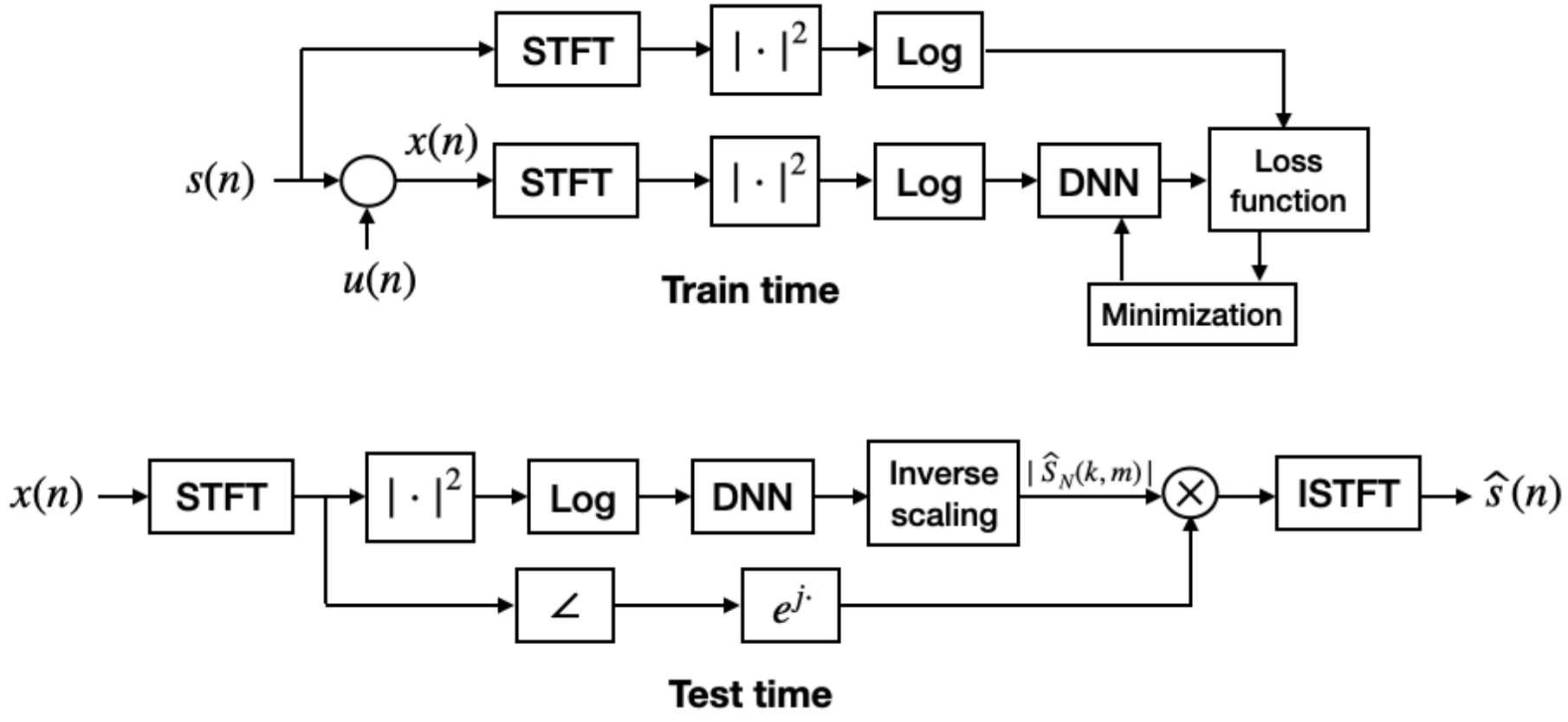
D. Wang & J. Chen, Supervised speech separation based on deep learning: An overview, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2018.

# DL-based speech enhancement: Direct spectrogram mapping



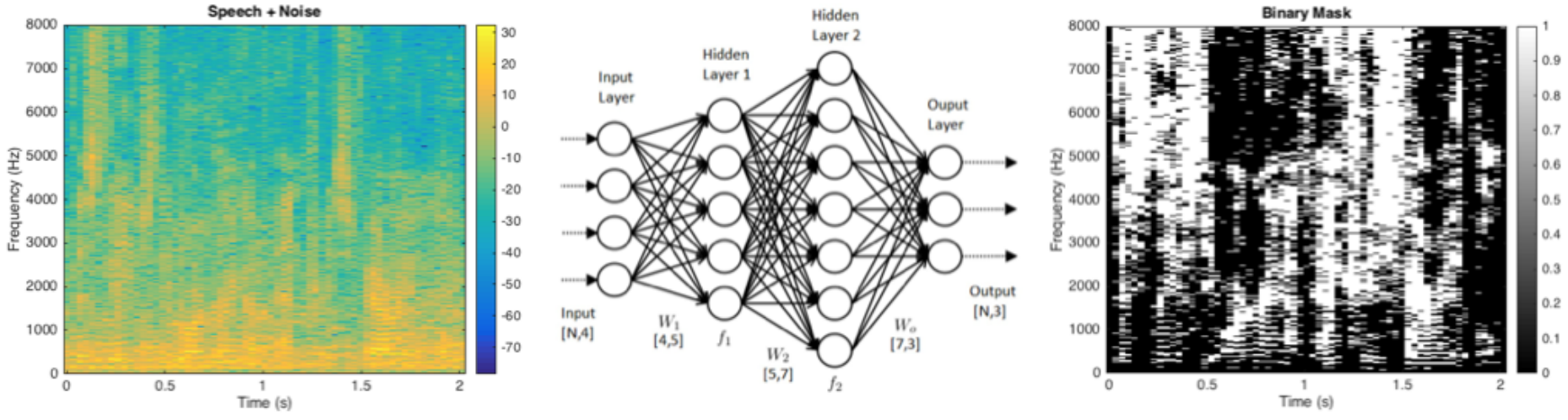

Note: The DNN schema is poorly chosen... In this application, input and output dimensions should match !!!

# DL-based SE: The direct-mapping complete pipeline




Note: This is a supervised method. At training time, we have examples of $s(n)$ and $u(n)$, though at test time we only have $x(n)$.

# DL-based speech enhancement: Spectrogram-to-mask mapping




Note: The DNN schema is poorly chosen... In this application, input and output dimensions should match !!!

# DL-based speech enhancement: Mask(s)

- Based on the sparsity of speech/audio in the TF domain = one source is predominant in each TF bin.
- Estimate a binary mask:

$$M(k,m) = \begin{cases} 1 & \text{if } X_N(k,m) \text{ is dominated by speech, i.e., } |S_N(k,m)|^2 > |U_N(k,m)|^2 \\ 0 & \text{otherwise} \end{cases}$$

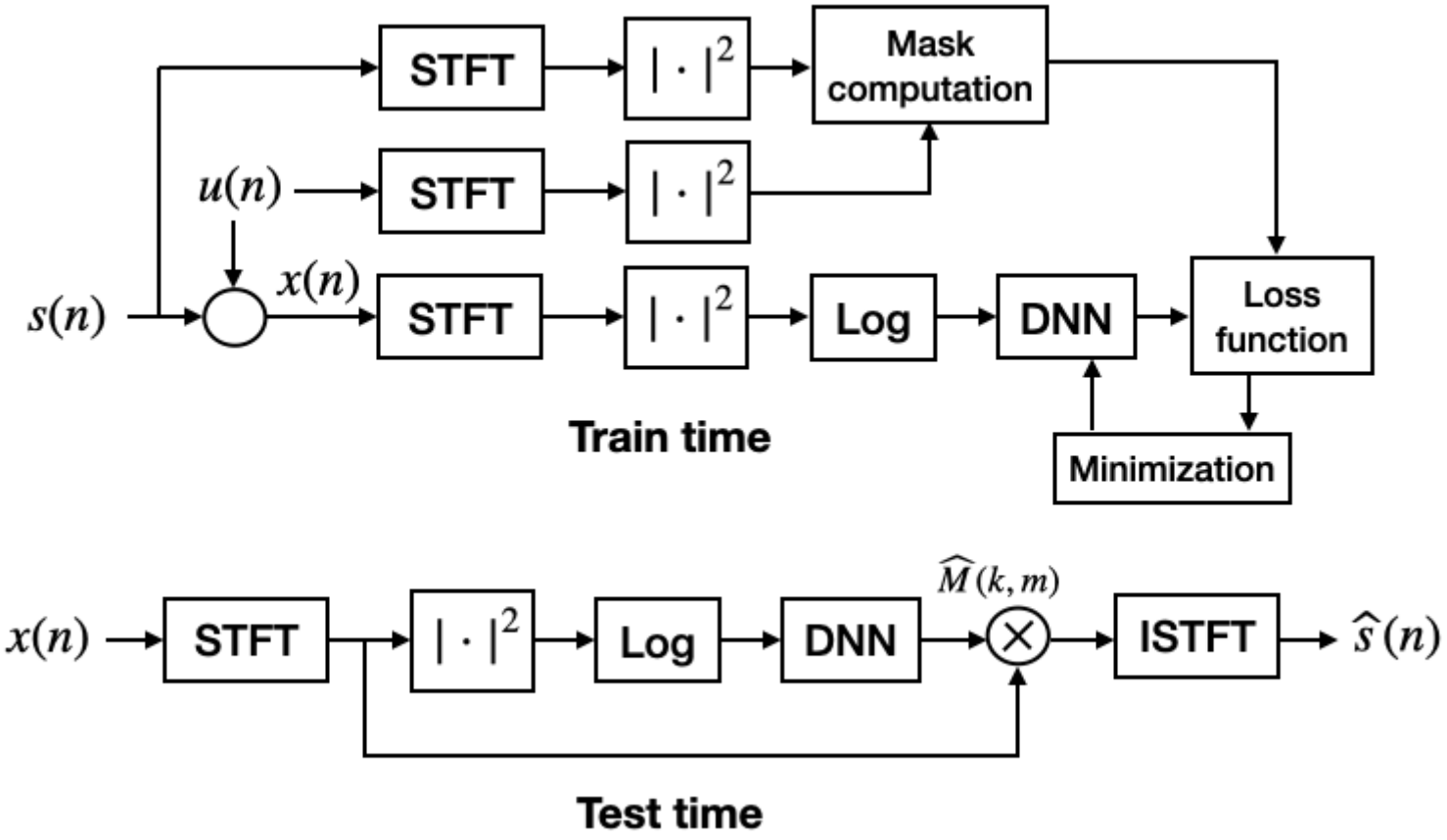
  It can be shown that this is the optimal binary mask in the MMSE sense.

- Extension to soft masks, $0 \leq M(k,m) \leq 1$ , which are strongly related to the Wiener filter.
- Then: $\widehat{S}_N(k,m) = M(k,m) \cdot X_N(k,m)$.



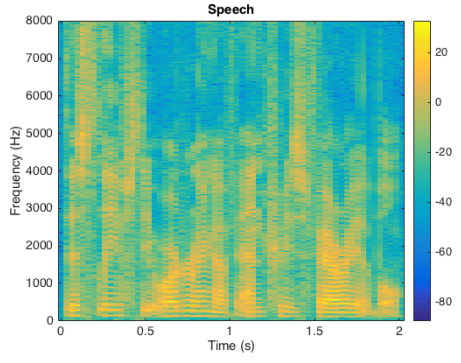
0. Yilmaz & S. Rickard, Blind separation of speech mixtures via time-frequency masking, IEEE Transactions on Signal Processing, 2004.

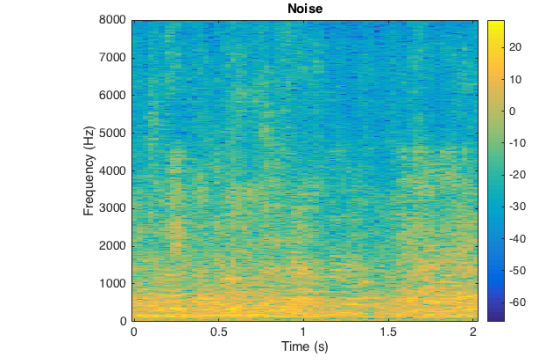
# DL-based SE: The mask-based complete pipeline

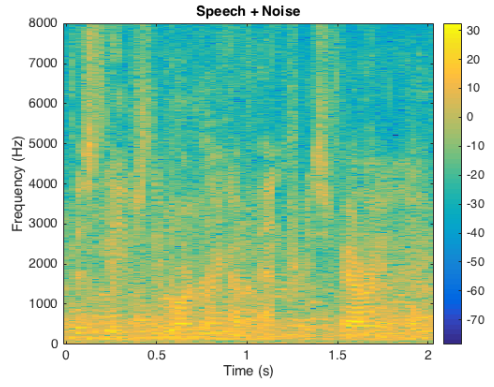


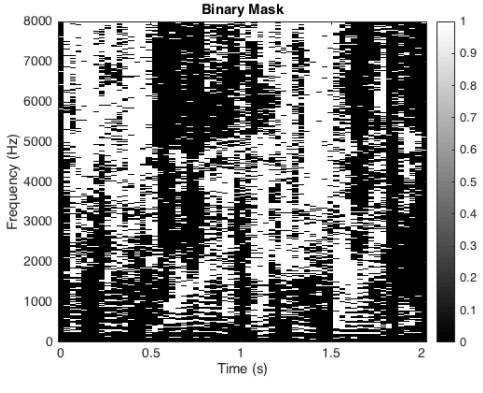

Note: Again, this is a supervised method.

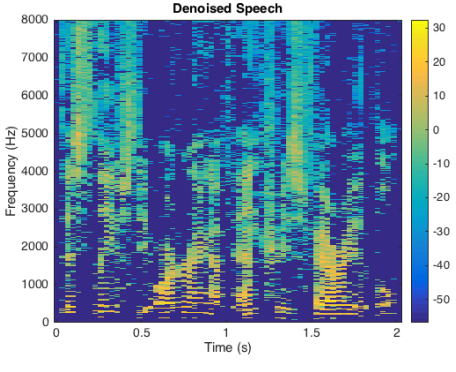
# Speech enhancement with Ideal Binary Masking (oracle)




- $\mathrm{SNR}_{\mathrm{in}} = 0$ dB
- Typically, $\mathrm{SDR}_{\mathrm{out}} \approx 15\mathrm{dB}$
- DNNs provide $8$ to $15\mathrm{dB}$ gains, depending on model complexity and dataset.

# DL-based SE: Complementary information

- A huge number of scientific papers within 2013-2018. See a review in (Wang & Chen, 2018).
- In addition to the two output strategies (spectrograms or masks), many studies considered:
  - different input features (in place of or complementary to the noisy spectrogram) and output features (e.g., masks different than binary or Wiener masks).
  - different DNN models: Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Convol. Recurrent Neural Networks (CRNNs), Residual Networks, etc. See the Deep Learning course and the Speech Processing course. It is best to have a model that exploit correlations efficiently in both the frequency and time dimensions.
  - different loss functions and training strategies.
- State-of-the-art performance for sophisticated models is a bit below oracle, but subject to train/test conditions mismatch.

D. Wang & J. Chen, Supervised speech separation based on deep learning: An overview, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2018.

# The DL Speech Enhancement project (projet de simulation logicielle)

If you do this project, you will have to implement a complete deep-learning-based single-channel speech enhancement system. You will mostly use Python. In a few more details, you will have to:

- Get familiar with a programming/test DL framework (e.g., Keras, PyTorch)
- Get familiar with basic signal processing routines (e.g., STFT)
- Download speech and noise datasets; generate noisy signals and organise your data into training/validation/test data
- Implement, train and evaluate different DNN models and their combinations (CNNs, RNNs, LSTMs, etc.)
- Evaluate the speech enhancement results with the SE metrics and with your own ears!
- Compare different data representations, different TF masking strategies, etc.
- Bonus: Search for other types of masks (there are better masks than Wiener!)

An advice: Start with a "simple" solution and make it more complex when things start to work.

# Complements: (Single-channel) Music unmixing

- Separation of the vocals or of any individual instrument from a music mix

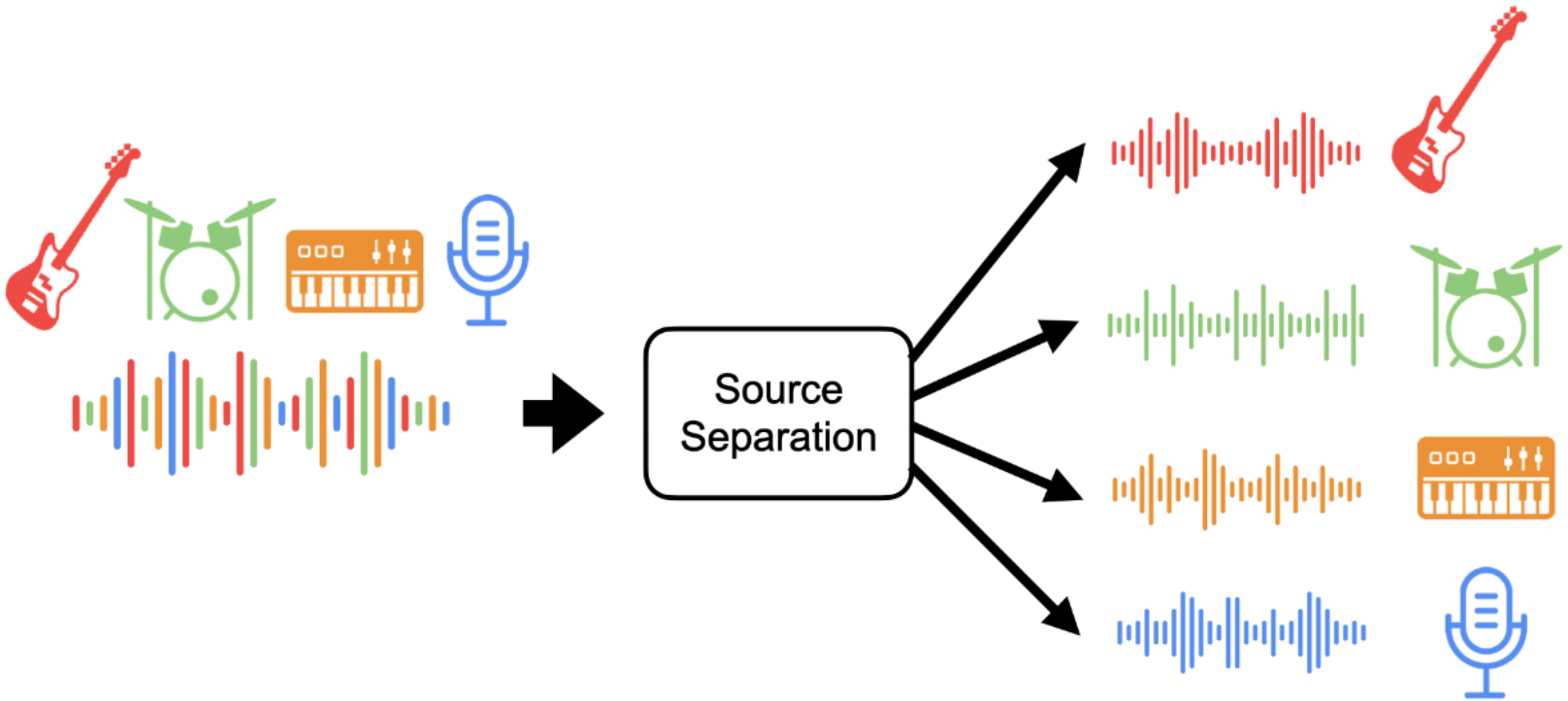



Image credit: E. Manilow, P. Seetharaman & J. Salamon, Open source tools & data for music source separation, online tutorial/book, https://source-separation.github.io/tutorial/landing.html.

# Complements: (Single-channel) Music unmixing

- We can use the same general supervised deep learning methodology as the one used for speech enhancement in noise: The source to extract from the mix is the "useful signal" (output of the DNN), all the rest is considered as the "noise", and the input is the mix signal.
- Basically, two main possibilities:
  - One model/training procedure for each type of instrument to separate
  - One single huge model with several different output sources; This is OK as long as (a) the sources to separate are of different nature, and (b) there is no instrument permutation during training!
- The key ressource here is data: Limited availability of open-access datasets of musical pieces with separated tracks (aka stems), though less and less limited...
- A lot of Web ressources (tutorials, interactive books, benchmark of methods, etc.)

https://sigsep.github.io/eusipco2019_tutorial/#/cover
https://source-separation.github.io/tutorial/landing.html
https://sisec18.unmix.app/#/
Z. Rafii et al., An overview of lead and accompaniment separation in music, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2018.

# Complements: (Single-channel) Speaker separation

- $x(n) = s_1(n) + s_2(n)$, with $s_1(n)$ and $s_2(n)$ being two speech signals.
  We want to obtain two estimates $\hat{s}_1(n)$ and $\hat{s}_2(n)$ from $x(n)$.

- Expected to be a more difficult problem than speech enhancement in noise, because the different sources in the mixture signal are of the same nature, and so are the expected outputs. How can a DNN model then rely on the spectral content to learn to separate them?

- Faces the source permutation problem: If $s_1(n)$ and $s_2(n)$ are of the same kind, why would a DNN output $s_1(n)$ on output channel $1$, and output $s_2(n)$ on output channel $2$, rather than $s_1(n)$ on output channel $2$ and $s_2(n)$ on output channel $1$?

- Yet it works, given that (a) you use a model that is able to exploit temporal information in spectrograms (some kind of Recurrent NN), and (b) if you use permutation invariant training.

- PIT: When computing the loss function and its gradient at training time, compute it for all possible sources permutation on the output channels, and keep the permutation that corresponds to the lowest loss function value.

M. Kolbæk et al., Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2017.

# Part 4: Spatial Audio

# Content of Part 4

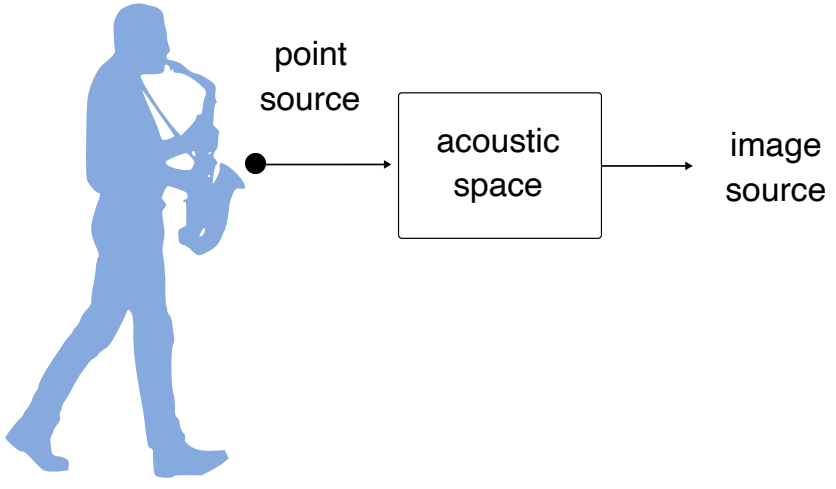
- Auralization
- Sound propagation in free-field and rooms
- Room impulse response
- Source separation based on spatial diversity

# The acoustic space

There can be no sound without a medium allowing for the propagation of acoustic waves.

Let us consider a sound source in a room (e.g., a speaker or musical instrument).

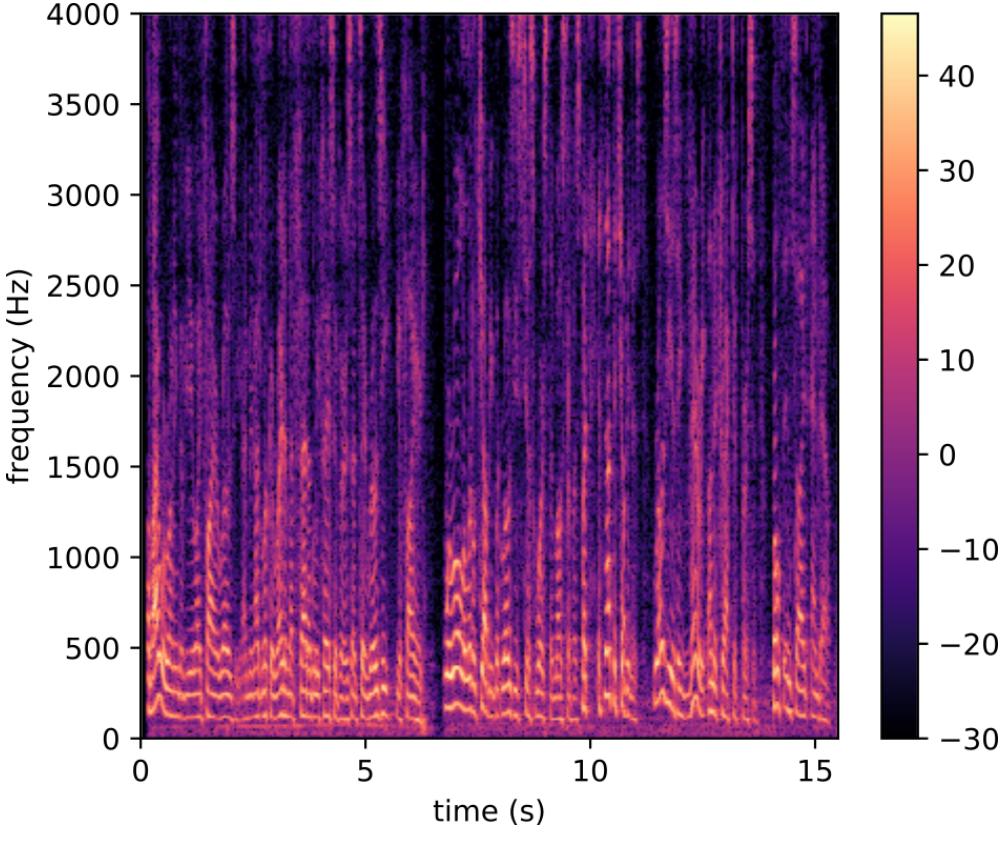
- The signal recorded by a microphone does not only characterize the sound source.
- It is shaped by the acoustic space, or acoustic channel, between the sound source and microphone position. It is also shaped by the microphone itself but we assume the influence of the microphone transfer function is negligible here.
- The resulting image source signal depends on the recording environment (e.g., room shape, wall materials), the source position, and the microphone position.



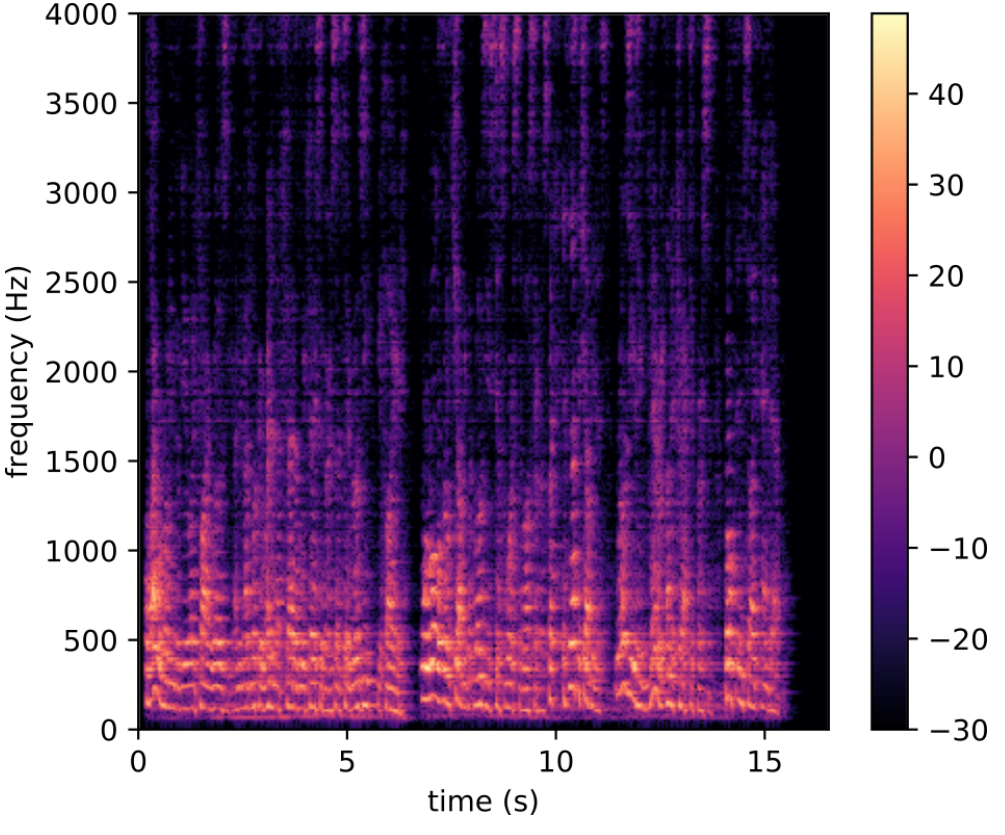

H. Kuttruff, Room acoustics, CRC Press, 2016.

# Example of room effect

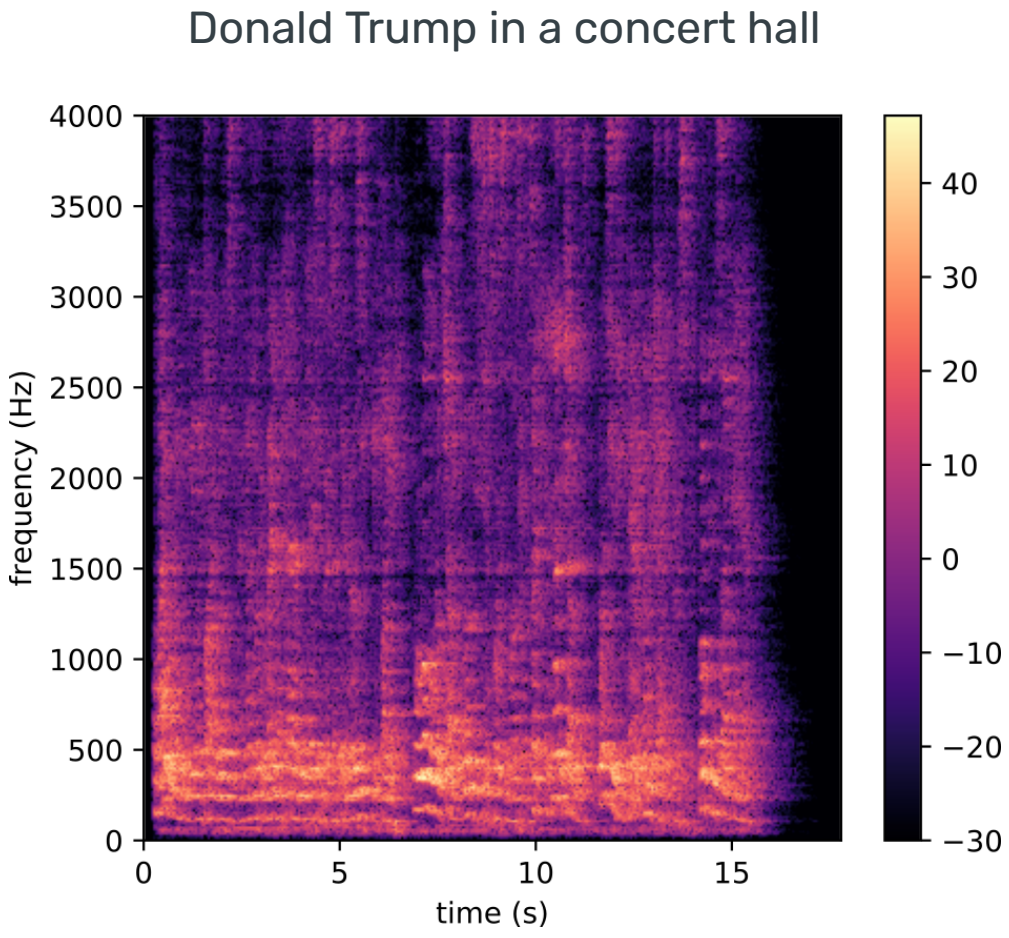
Original recording of Donald Trump




Donald Trump in a bathroom

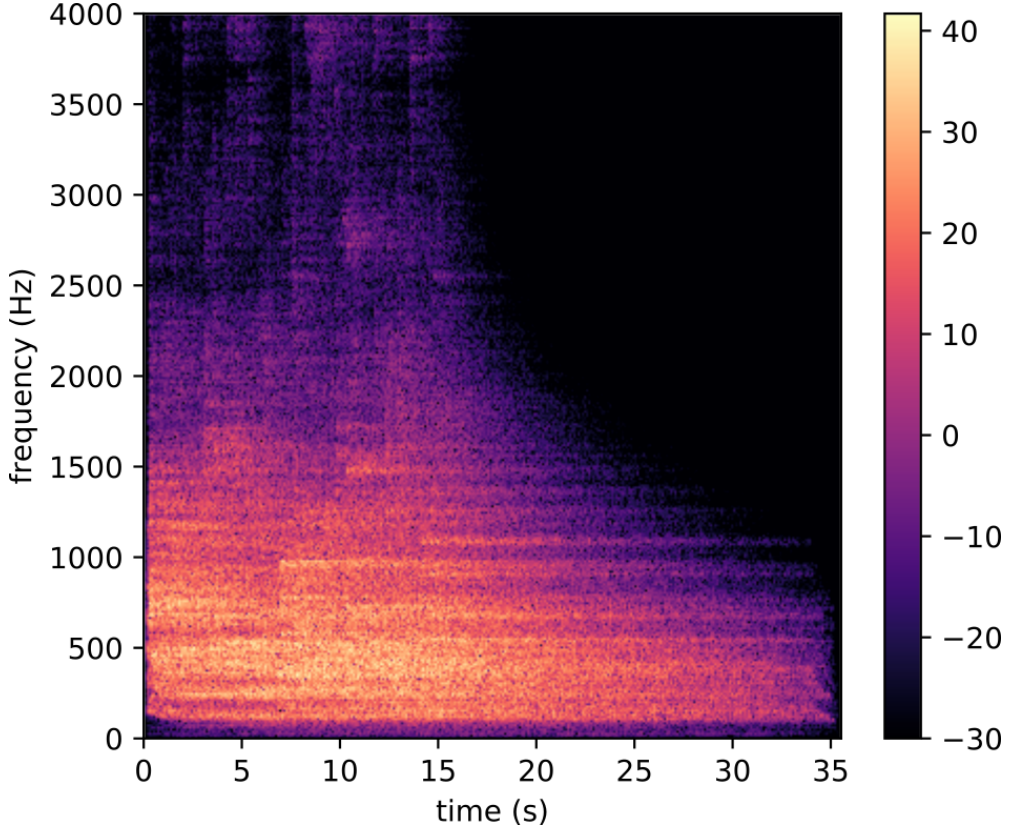


# Example of room effect

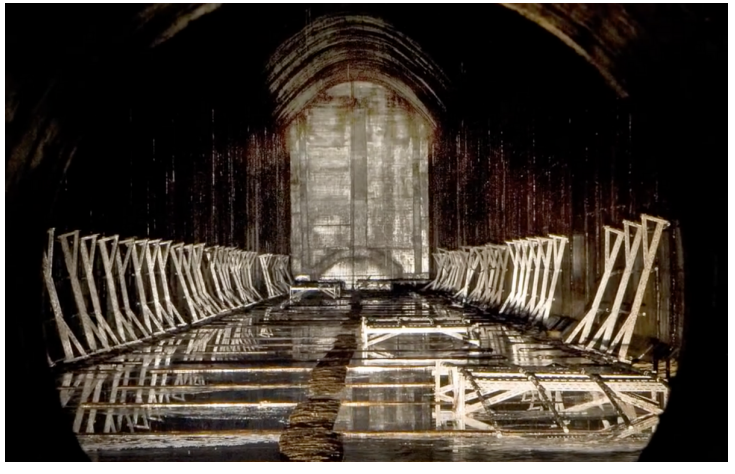


# Example of room effect

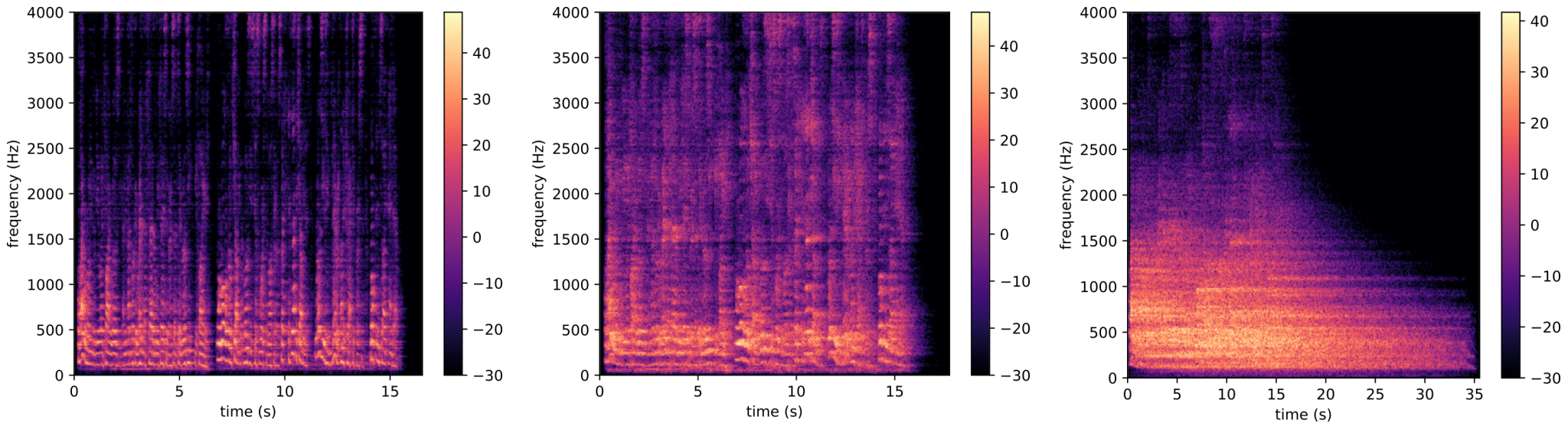
Donald Trump trapped in the Inchindown oil tanks in Scotland




World's longest reverberation



Images credit: Wikipedia




- The acoustic space has a great influence on the recorded audio signal.
- For example, reverberation may affect the intelligibility of a speech signal.

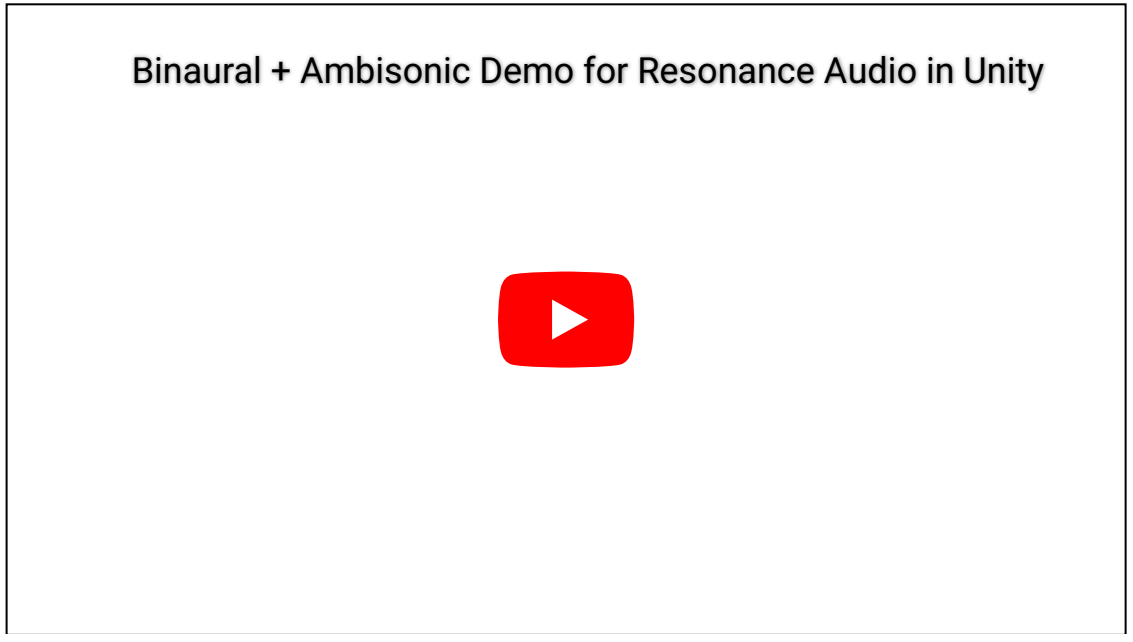
  This is not a problem in the case of Donald Trump's hate speech, but generally speaking it makes extraction of information challenging for machine listening systems.

Understanding the effects of the acoustic space on audio source signals is important

- to build robust machine listening systems,
- to design hearing aid systems,
- to design a concert hall (and nice houses to live in) in architectural acoustics,
- to design anti-noise systems (e.g., acoustic barriers),
- to synthesize virtual acoustic environments in video games,
- etc.

To create a truly immersive virtual reality experience, all our senses have to be involved. Being able to hear spatial sounds that match with what we see and how we move is very important.

*3D audio demo of the Resonance Audio project developed by Google (use earphones or a headset).*




Listen how the spatial characterisitics of acoustic sources change as the player moves, and how different rooms have different acoustics. And find out a huge mistake!

# Auralization

We did not actually lock Donald Trump in the Inchindown oil tanks. The previous examples used the principle of auralization.

> Auralization is the process of creating a **virtual rendition of a sound in a space.**
>
> An auralized sound corresponds to the **convolution** of an **anechoic sound** with a **room impulse response**.

- Anechoic sound: a sound recorded in an acoustic environment with (almost) no reflections.
- Room impulse response (RIR): characterizes the acoustic path between two points in a room, including the reflections on the walls, floor, ceiling, objects, etc.

# Convolution reminder

The convolution between two signals $u(n)$ and $v(n)$ is defined for all $n \in \mathbb{Z}$ by:

$$u(n) \star v(n) = [u \star v](n) = \sum_{p \in \mathbb{Z}} u(p)v(n-p)$$
$$= \sum_{p \in \mathbb{Z}} v(p)u(n-p).$$

Linearity:

$$[u \star (\alpha v + \beta\, w)](n) = \alpha[u \star v](n) + \beta[u \star w](n)$$

Time-invariance:

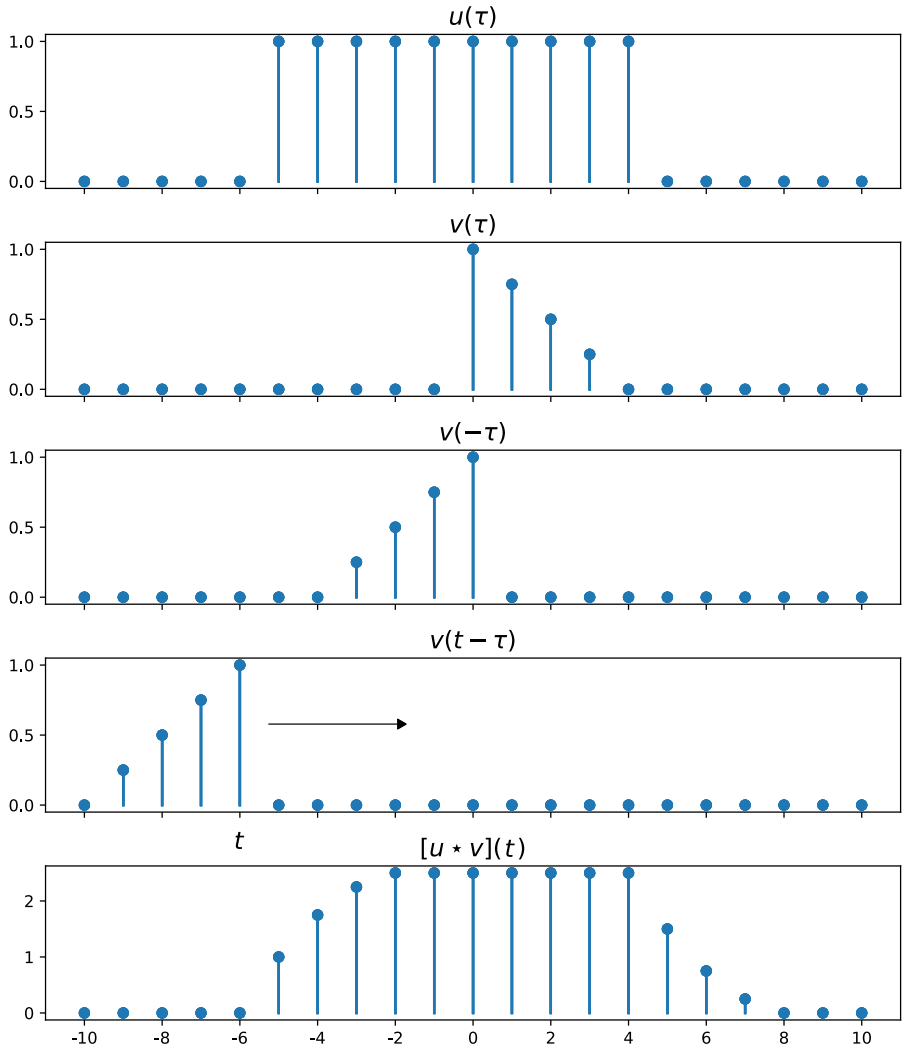
$$u(n) \star v(n-p) = [u \star v](n-p)$$

Commutativity:

$$[u \star v](n) = [v \star u](n)$$

Associativity:

$$[(u \star v) \star w](n) = [u \star (v \star w)](n)$$



# Sound propagation in free field

Before studying auralization in rooms, let us focus on sound propagation in free field.

# Free field

**The free field is an idealized situation in which there are no sound reflections.**

- In nature (open air environment), free field conditions can only be found if there is no close obstacle and sound reflections on the floor can be neglected, e.g. in new snow.
- Free field conditions can be artificially produced in anechoic chambers.

  They are used for acoustic measurements and sound perception experiments, as results are only influenced by the sound of the source and not by the properties of the room.

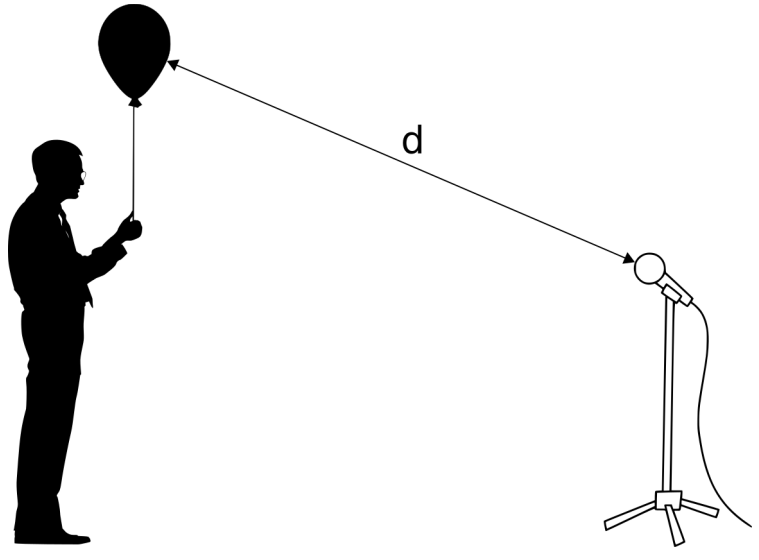
*Anechoic chamber at IRCAM, Paris.*
*Image credits: Philippe Barbosa, Le Monde.*

# Source-to-microphone propagation in free field

- Let us consider a man holding a balloon and a microphone in an open air environment without any obstacle.

  *We assume the man and the microphone are not reflecting sound.*

- The baloon explodes, it produces a (analog) source signal $s_0(t)$.




How is the source signal $s_0(t)$ related to the microphone signal $x_0(t)$?

The signal arriving at the microphone is given by

$$x_0(t) = \frac{1}{\sqrt{4\pi}d} s_0\left(t - \frac{d}{c}\right)$$

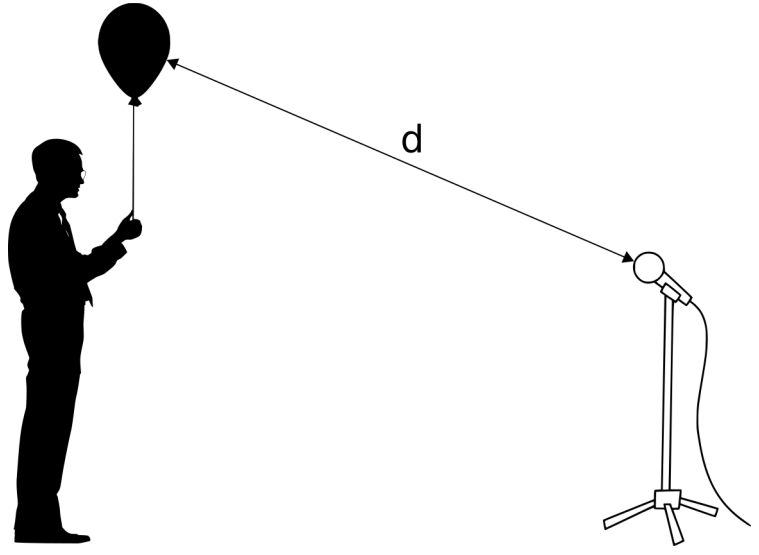
where

- $d$ is the source-to-microphone distance (in m);
- $c = 343$ is the speed of sound (in m/s at 20°C);
- $d/c$ is the time of arrival (in s);

The digital version of the microphone signal is given by

$$x(n) = \frac{1}{\sqrt{4\pi}d} s\left(n - \frac{d}{c}F_s\right)$$

where $F_s$ is the sampling rate (in Hz) (neglecting sampling + quantization issues).



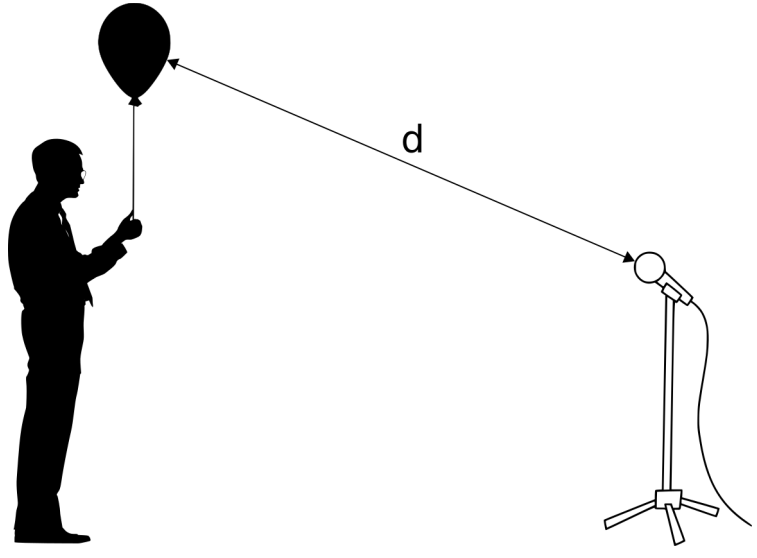

At the microphone, the source signal is simply **attenuated** and **delayed**. This is an **anechoic recording**.

We can rewrite the microphone signal as:

$$x(n) = [h \star s](n)$$

where

$$h(n) = \frac{1}{\sqrt{4\pi}d}\delta\left(n - \frac{d}{c}F_s\right), \qquad \delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}.$$

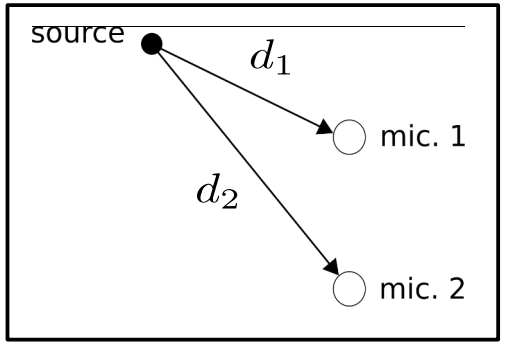



$h(n)$ characterizes the acoustic path between the source and the microphone.

# Multi-microphone setting

- In a multi-microphone recording, we have different attenuations and delays at each microphone $i$:

$$x_i(n) = \frac{1}{\sqrt{4\pi} d_i} s\left(n - \frac{d_i}{c} f_s\right)$$




- Combining the expressions of $x_1(n)$ and $x_2(n)$, we can write:

$$x_2(n) = \boxed{\frac{d_1}{d_2}} x_1\left(n - \boxed{\frac{d_2 - d_1}{c}} f_s\right).$$

$\frac{d_1}{d_2}$ is the **level ratio** and $\frac{d_2 - d_1}{c}$ is the **time difference of arrival** (TDoA). These two features convey information about the **source position**.

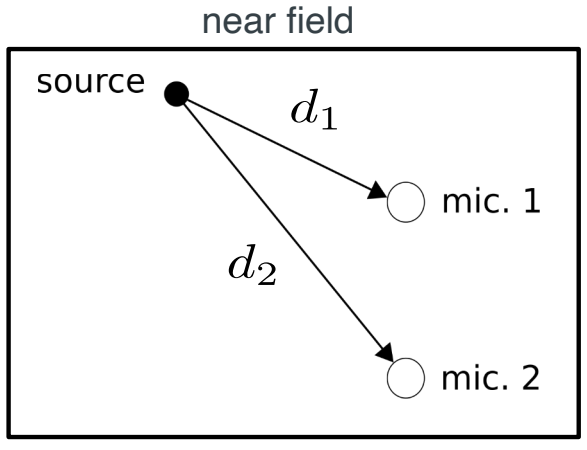
# Far-field case - Introducing sound source localization

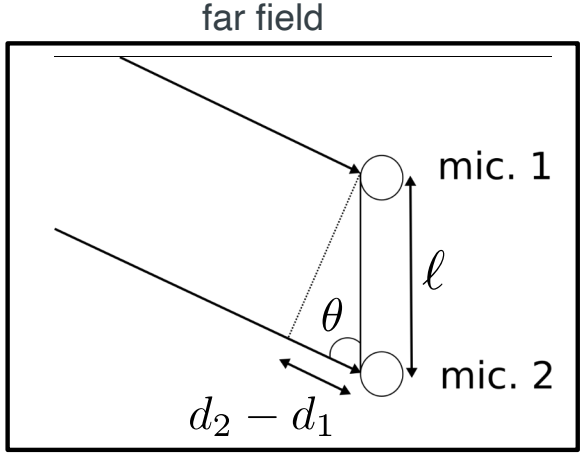
Assuming the source-to-microphone distances $d_i$ are large compared with the inter-microphone distance $\ell$:

- The level ratio is almost equal to one: $\frac{d_1}{d_2} \approx 1$.
- The TDoA is fully determined by the direction of arrival (DoA) and corresponding DoA angle $\theta$:

$$\frac{d_2 - d_1}{c} = \frac{\ell}{c} \cos(\theta).$$

- A measure of the TDoA (using, e.g., cross-correlation) can provide an estimate of $\theta$. This is one of the bases of (automatic) sound source localization (SSL).
- The more microphones we have, the more inter-channel TDoA measures we can make, the more robust the SSL (in noise for example).



# Sound propagation in rooms

# Room impulse response

The room can be considered as a linear time-invariant causal system (if we neglect the changes in temperature, pressures, etc.) so we still have

$$x(n) = [h \star s](n),$$

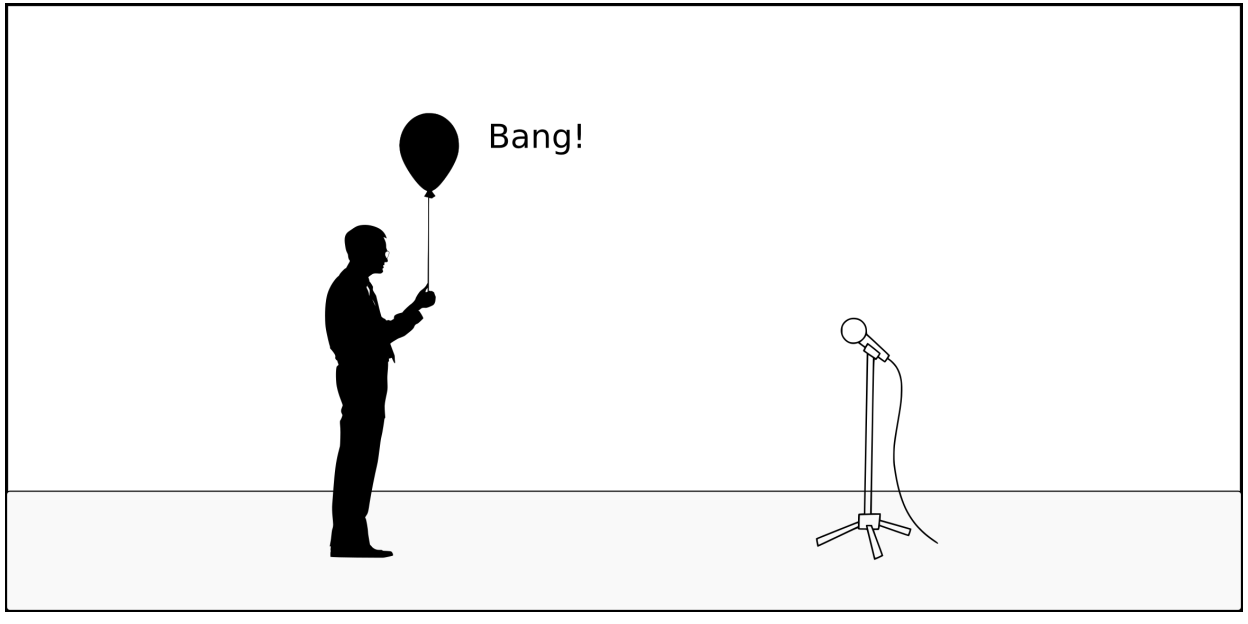
where $h(n) = 0$ for $n < 0$ due to causality.

$h(n)$ is called the **room impulse response** (RIR). It is defined for every couple of source and microphone positions. It completely characterizes the acoustic path between these two positions in the room.

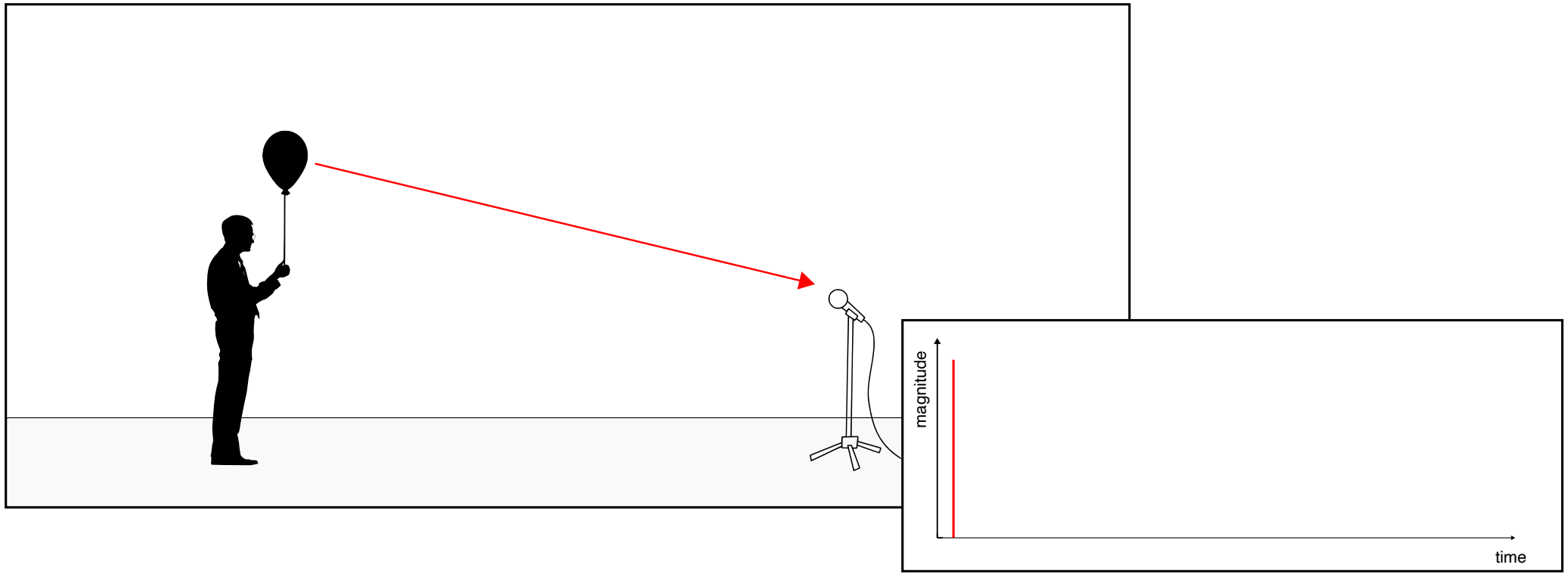
The source signal is filtered by the RIR, resulting in the recorded image source signal.

$h(n)$ is called the room *impulse* response because it is the response of the room when the excitation is an impulse (Dirac delta function):

$$x(n) = [h \star \delta](n) = h(n).$$

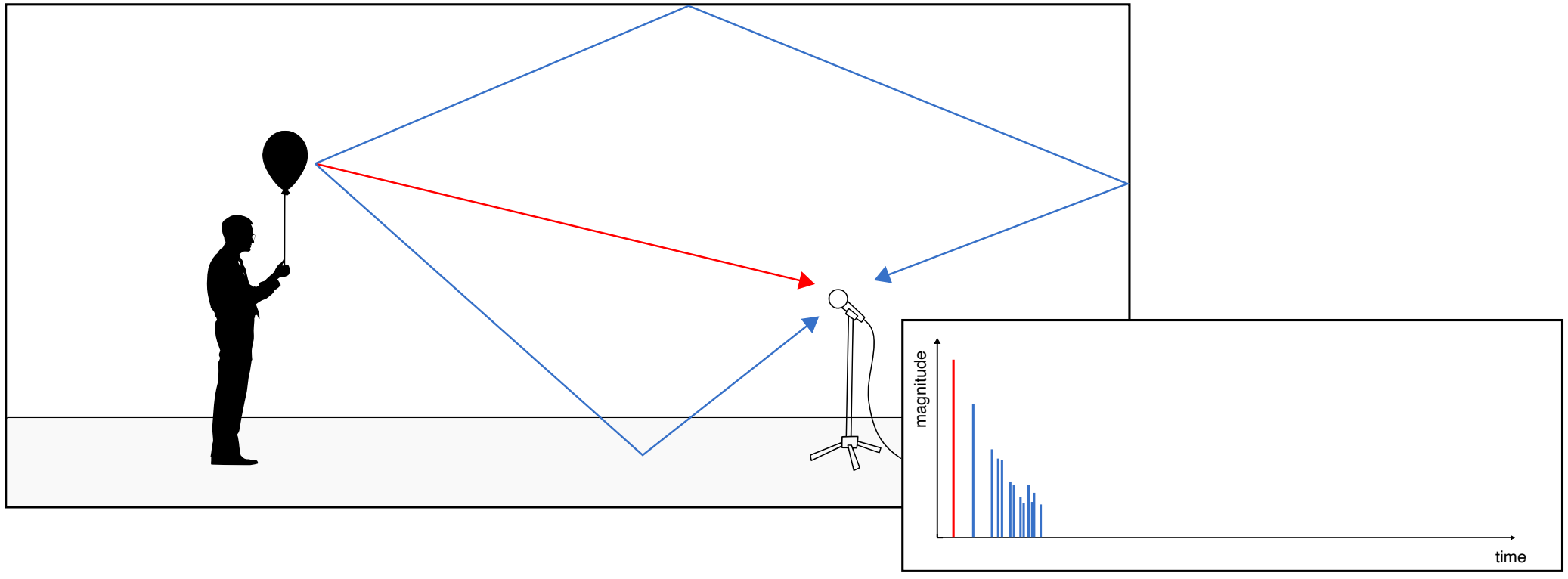


# Schematic illustration of an RIR



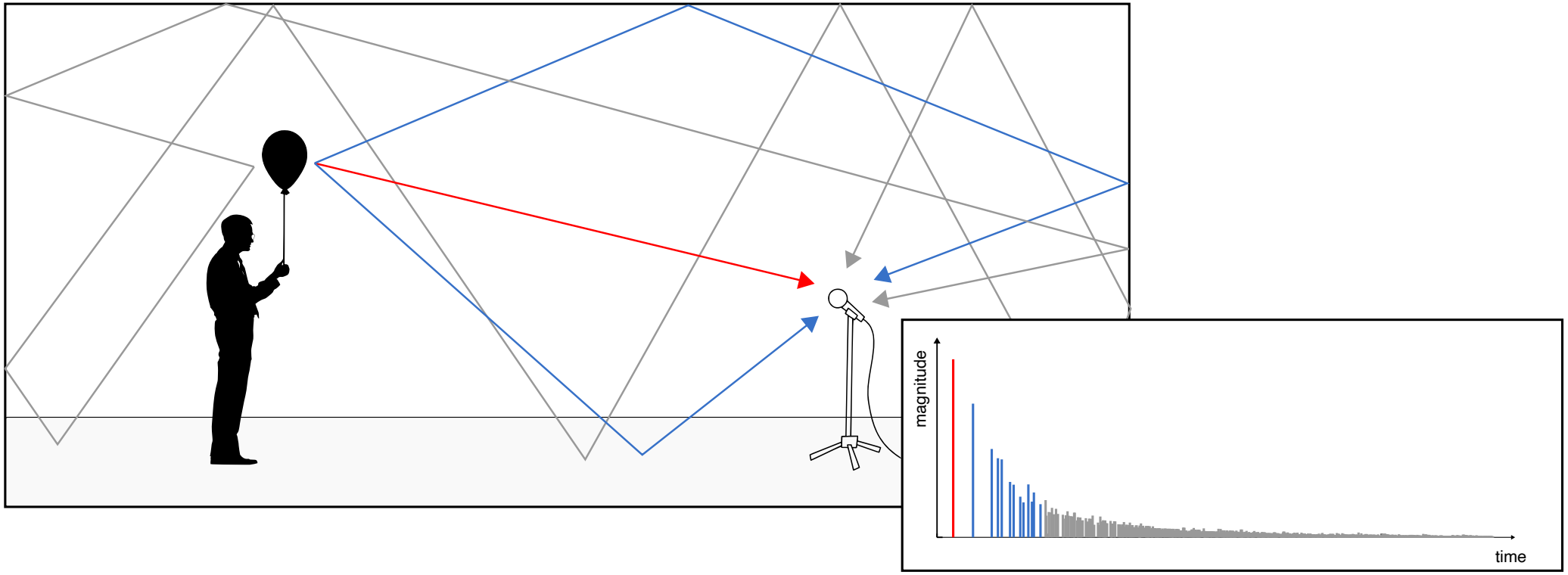

The direct path is... the direct path between source and microphone, i.e., in the line of sight. The direct-path component $x(n)$ is the first version of $s(n)$ to arrive at the microphone (similar to free-field propagation).

# Schematic illustration of an RIR




Then come the first significant reflections (e.g. on the floor) called the early echoes.

# Schematic illustration of an RIR




After some time, echoes have bounced many times on the room surfaces, and we have so many reflections that arrive simultaneously that we cannot distinguish them individually, this is called late reverberation.

When a source emits sound in a room, many successive reflections typically occur before the sound power becomes negligible. This is called **reverberation**.

Reverberation is induced by **multiple propagation paths between the source and the microphone**, each with its own delay and attenuation factor.

# RIR model and real RIRs

- In the above simple RIR model, we only consider specular reflections that are each characterized by an attenuation factor and a delay.
- In real rooms, sound propagation is more complicated than this.

  When a sound wave propagates in a room, it encounters surfaces with which it interacts in different ways:

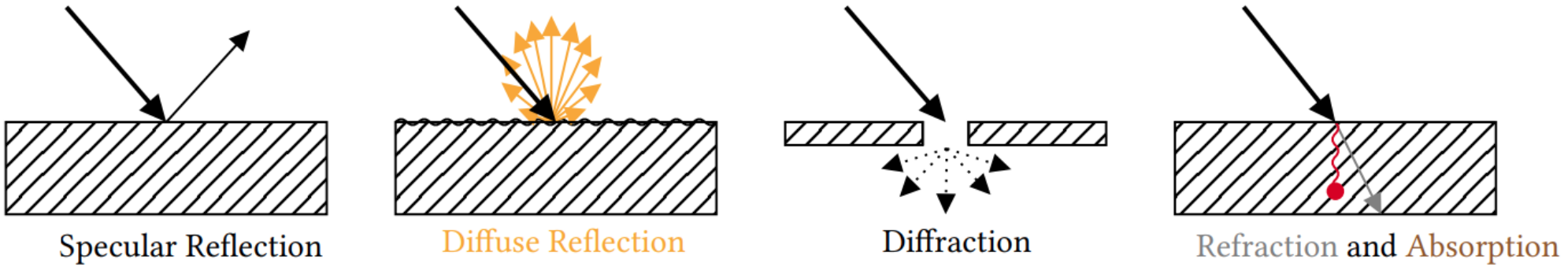
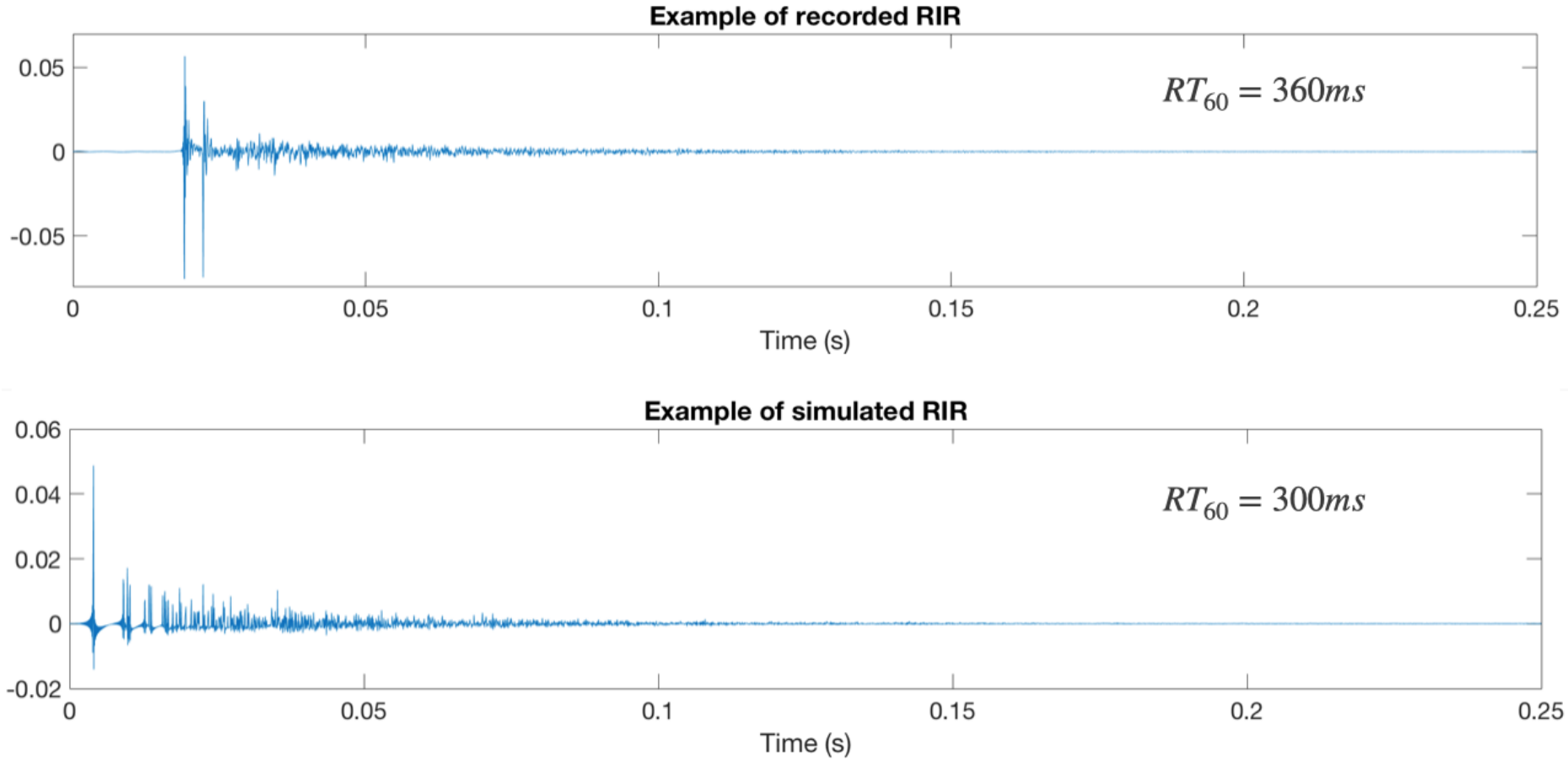



Image credits: D. Di Carlo, Echo-aware signal processing for audio scene analysis, Ph.D. Thesis, 2020.

# An example of recorded and simulated RIR




RIR dataset from Bar-Ilan Univ. / RWTH Aachen Univ.: E. Hadad et al., Multichannel audio database in various acoustic environments, International Workshop on Acoustic Signal Enhancement, 2014.
Dataset available at https://www.eng.biu.ac.il/gannot/downloads/

RIR generator from Audio Labs Erlangen: E. Habets, Room impulse response generator, Technical Report, 2006.
Software available at https://www.audiolabs-erlangen.de/fau/professor/habets/software/rir-generator.

# Simulating RIRs

- We can simulate RIRs based on the geometry of the room.
- The simplest method is called the image source method. It uses the previous simple attenuation + delay model.
- Below is a simulation of the acoustics in the Church of the Redentore in Venice.

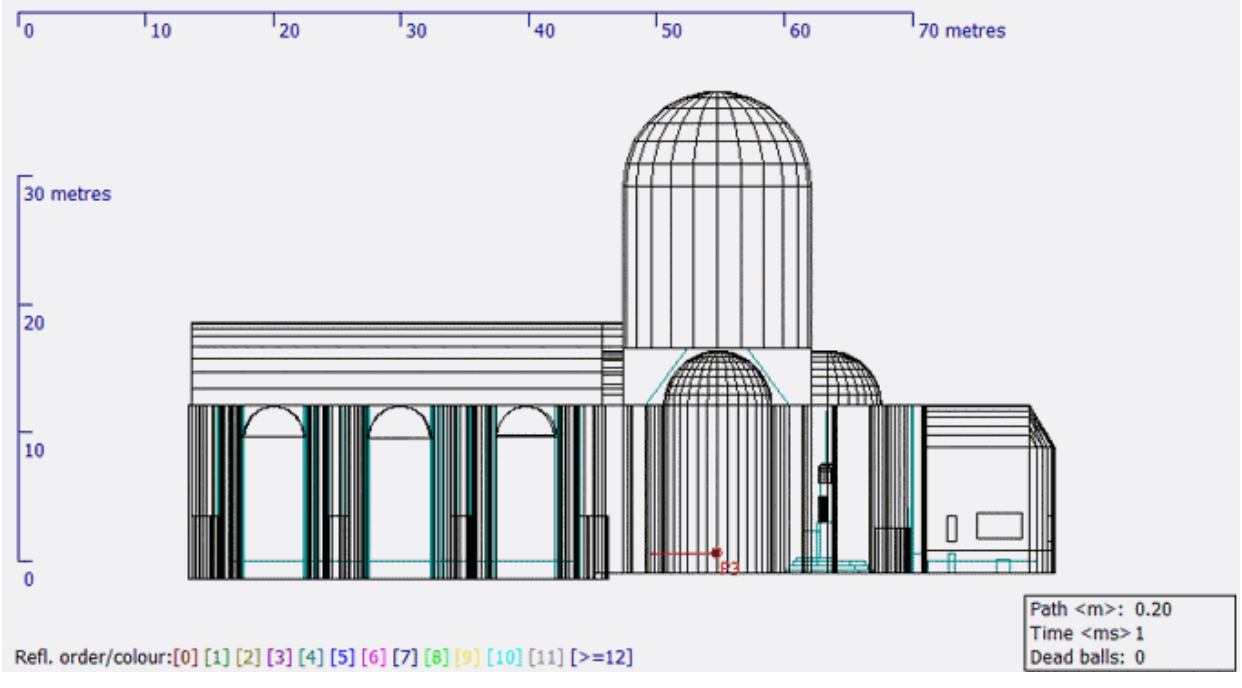
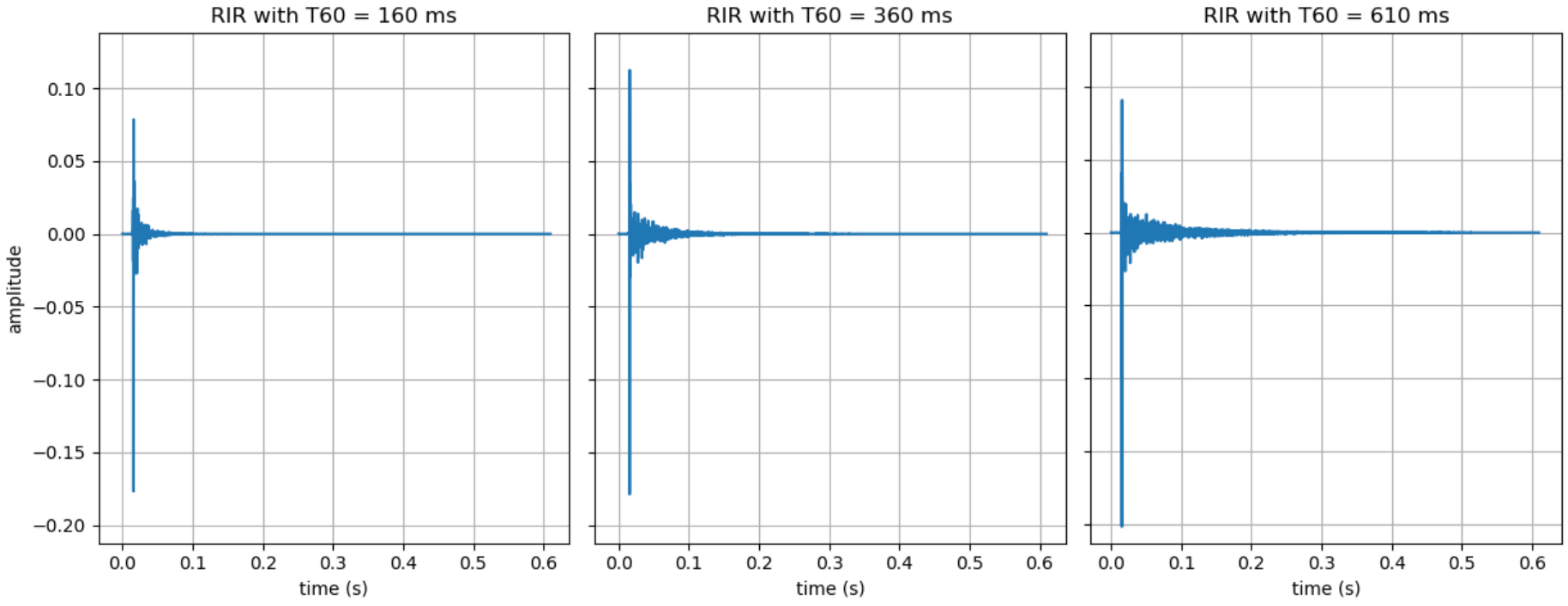



Image credit: B. Boren/American Univ.

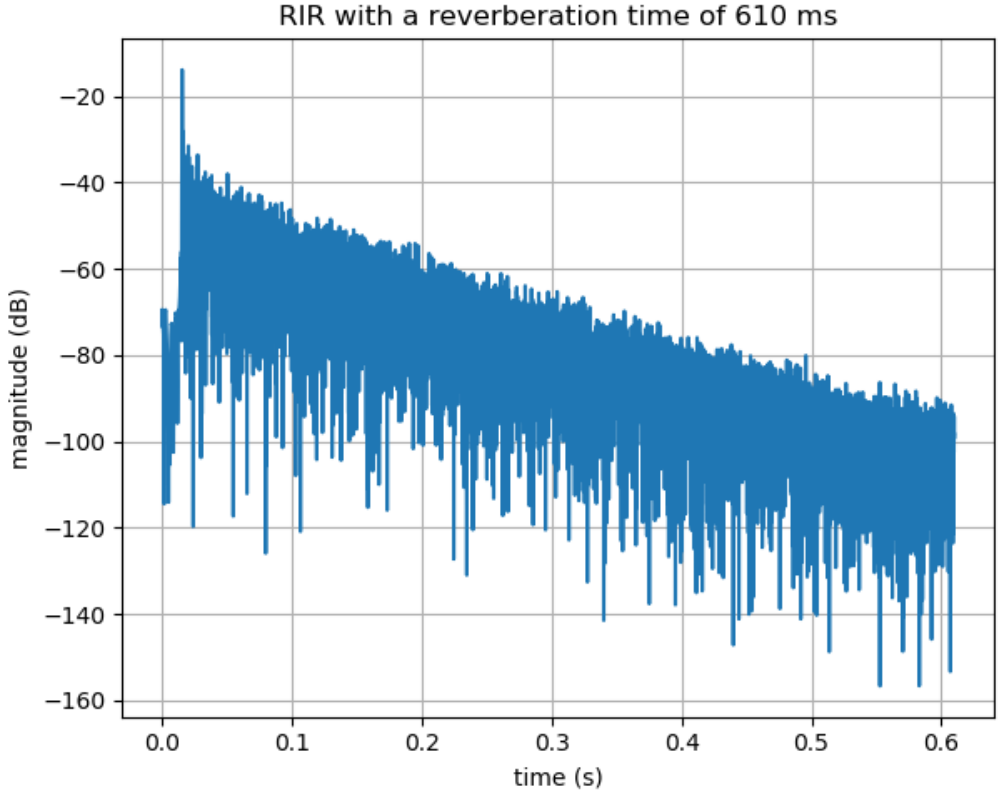
# Reverberation time




- These RIRs were measured in a room with fixed source and microphone but with a configurable reverberation level (with panels made of different materials placed on the walls), resulting in three different reverberation times (RTs).

# Reverberation time

- RT is the time it takes for the energy of a "short" sound source to decrease by $60\mathrm{dB}$.
- It depends on the room characteristics (mostly its size and the wall/floor/ceiling materials). The larger the room, the longer the reverberation time (with similar materials).



# Direct-to-reverberant ratio

- The direct-to-reverberant ratio (DRR) of an RIR or image signal is the ratio between the energy of the part of the RIR/signal that corresponds to the direct-path propagation and the energy that corresponds to the reverberation part (here including both early echoes and late reverberation).
- In general, the closer the source to the microphone, the higher the DRR. Also, for a good DRR, the microphone should better not be close to the walls.
- DRR and reverberation time are the two main indicators of the amount of reverberation (and thus of the difficulty to dereverberate the signal for example).

Brian Katz took detailed acoustic measurements in Notre Dame in 2013, using microphones on stands and a dummy head. These benchmark data will help reveal how rebuilding could alter the cathedral's sound.

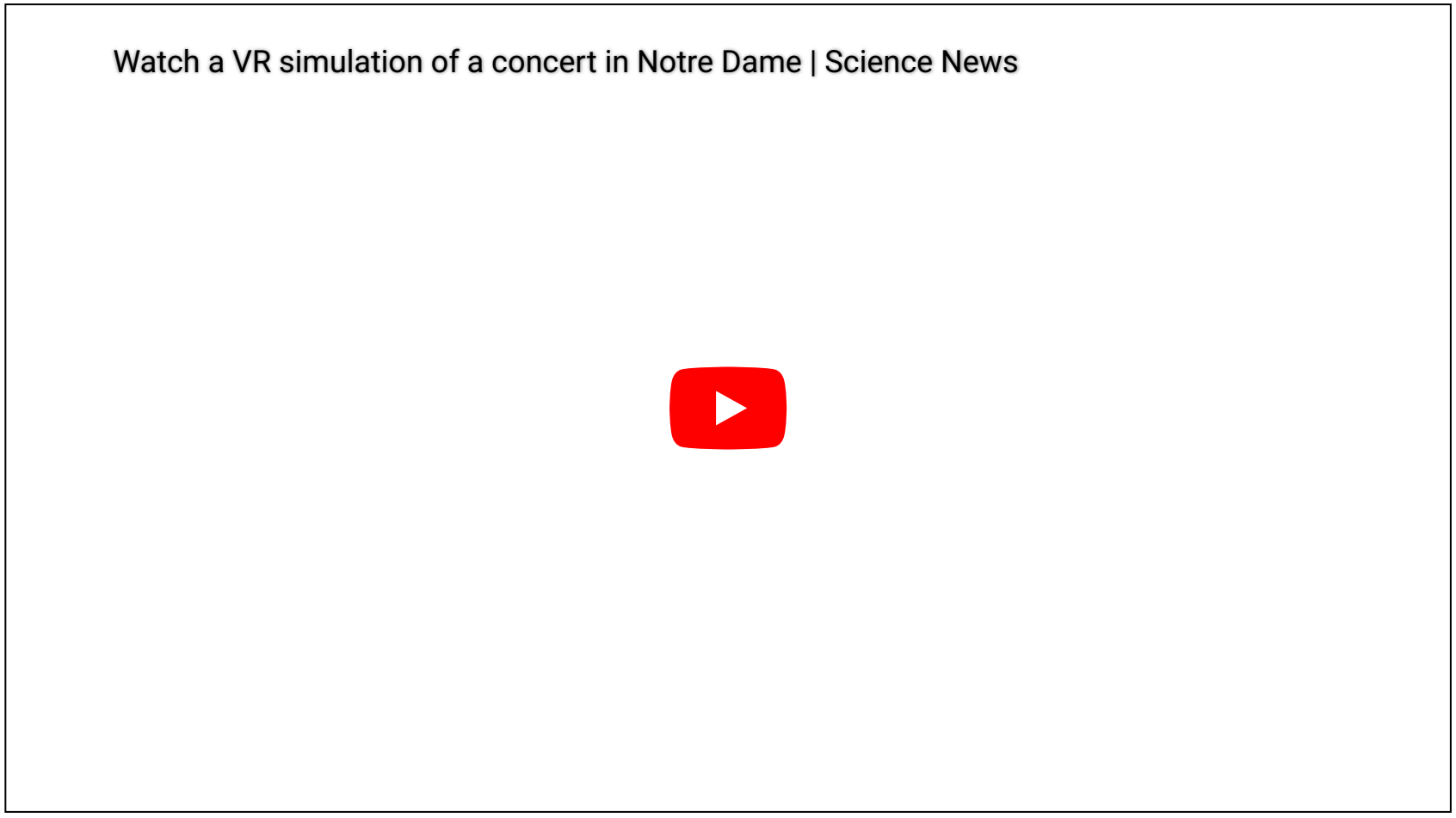
B. KATZ/CNRS

In July, three months after the fire, Mylène Pardoen (shown, left), Brian Katz and others wore protective suits and breathing masks inside Notre Dame. Fallen masonry and charred timbers littered the cathedral's floor.

BOTH: B. KATZ AND M. PARDOEN/CNRS

Brian Katz, a CNRS researcher, measured RIRs in Notre Dame de Paris in 2013. These data may help restoring the acoustics of the cathedral.

More info: https://www.sciencenews.org/article/notre-dame-cathedral-fire-legendary-acoustics-restoration




A virtual reality simulation of a concert in Notre Dame, a work-in-progress by Brian Katz and colleagues, re-creates the acoustics of the church. In this clip from the 360-degree video, the sound changes as the viewer moves around the cathedral.

Video: Brian Katz and David Poirier-Quinot/CNRS, Sorbonne University

# Source separation exploiting the spatial diversity and the spectral sparsity of sources in anechoic acoustic recordings

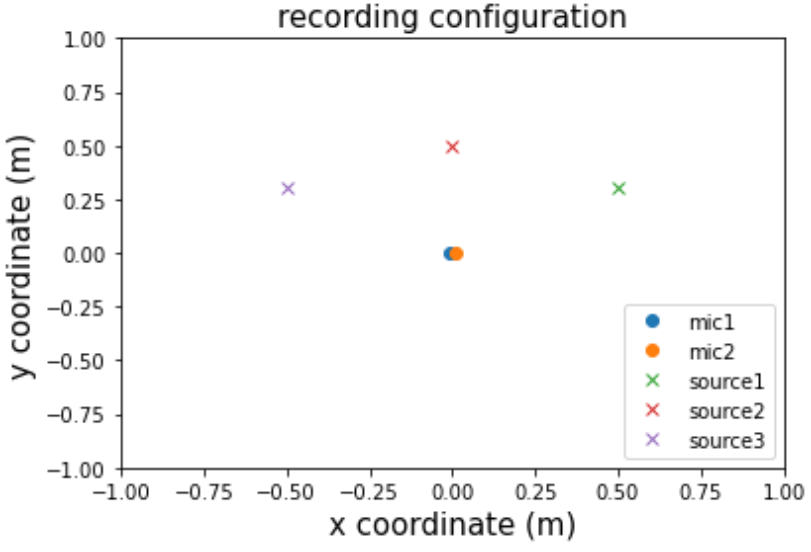
The Degenerate Unmixing Estimation Technique (DUET) (Yilmaz & Rickard, 2004)



O. Yilmaz & S. Rickard, Blind separation of speech mixtures via time-frequency masking, IEEE Transactions on Signal Processing, 2004.

# Unmixing or source separation

We want to estimate individual speech source signals from an anechoic stereophonic (2-channel) mixture.




If we have more sources than sensors (here 2 microphones), the problem is under-determined or degenerate (it is not "invertible" mathematically).

# Anechoic stereo mixture model in the time domain

Without loss of generality, we can absorb the attenuation and delay parameters at the first microphone into the definition of the source signal (i.e., a change of variable; we now call $s_j(n)$ the $j$-th source signal at microphone $1$).

$$x_1(n) = \sum_{j=1}^{J} s_j(n), \qquad x_2(n) = \sum_{j=1}^{J} a_j s_j(n - \delta_j),$$

where

- $a_j = d_{1j}/d_{2j}$ is the level ratio (or relative attenuation factor) between the microphones for the $j$-th source.
- $\delta_j = \dfrac{d_{2j} - d_{1j}}{c} F_s$ is the TDoA between the microphones for the $j$-th source.

In the following, we will refer to $\{(a_j, \delta_j)\}_{j=1}^{J}$ as the mixing parameters.

# Anechoic stereo mixture model in the STFT domain

Assuming that the TDoAs are small compared to the STFT analysis window length $N$, we have (here, we omit $N$ as a subscript of the STFT for clarity):

$$s_j(n-\delta_j) \overset{\text{STFT}}{\longleftrightarrow} \exp\left(-i2\pi\frac{k\delta_j}{N}\right) S_j(k,m).$$

The mixture model thus rewrites in STFT domain as follows:

$$X_1(k,m) = \sum_{j=1}^{J} S_j(k,m), \qquad X_2(k,m) \quad = \sum_{j=1}^{J} a_j \exp\left(-i2\pi\frac{k\delta_j}{N}\right) S_j(k,m),$$

or in matrix form: $\begin{bmatrix} X_1(k,m) \\ X_2(k,m) \end{bmatrix} =$

$$\begin{bmatrix} 1 & \dots & 1 \\ a_1 \exp\left(-i2\pi\frac{k\delta_1}{N}\right) & \dots & a_J \exp\left(-i2\pi\frac{k\delta_J}{N}\right) \end{bmatrix} \begin{bmatrix} S_1(k,m) \\ \vdots \\ S_J(k,m) \end{bmatrix}$$

# DUET principle

It is possible to separate $J > 2$ sources from **the stereo anechoic mixture** provided that

- the **time–frequency representations of the sources do not overlap** (assumption 1),
- the sources have **different spatial locations** (assumption 2).

# W-disjoint orthogonality (assumption 1)

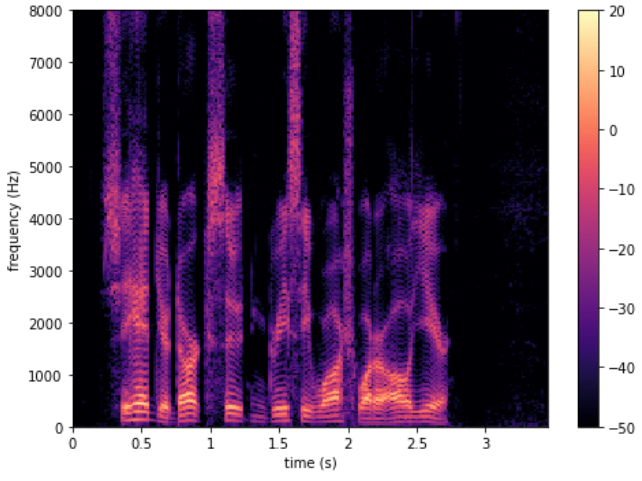
- Speech/music/audio source signals are assumed to have disjoint time-frequency supports, a.k.a. the W-disjoint orthogonality (WDO) hypothesis. In other words, at most one source is assumed to be active at each time-frequency point $(k, m)$.
- This can be formalized by: $S_j(k,m)S_l(k,m) = 0, \ \ \forall (k,m), \ \ \forall j \neq l.$
- The mixture model simplifies as follows:

$$\begin{bmatrix} X_1(k,m) \\ X_2(k,m) \end{bmatrix} = \begin{bmatrix} 1 \\ a_{\mathcal{I}(k,m)} \exp\left(-i2\pi \frac{k\delta_{\mathcal{I}(k,m)}}{N}\right) \end{bmatrix} S_{\mathcal{I}(k,m)}(k,m).$$

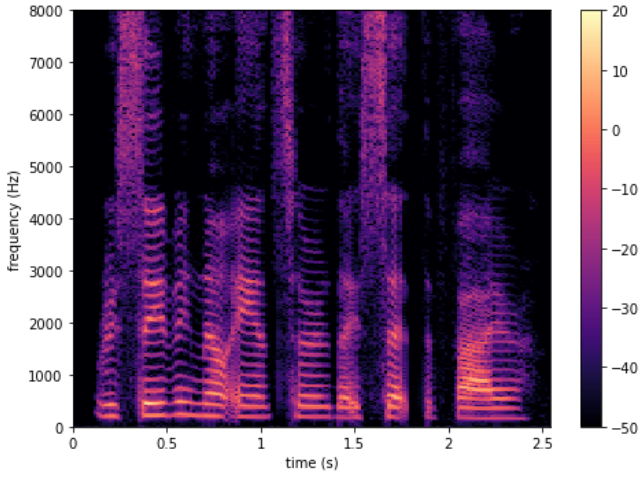
  where $\mathcal{I}(k,m) \in \{1, 2, ..., J\}$ indicates which source is active at TF point $(k, m)$.

- In practice, even if this is an "idealized" view of audio mixtures, most TF points in a mixture are dominated by one of the sources, i.e., the energy of that source is much larger than the energy of any other source. This is sufficent to ensure good separation using the above model. We talk about speech/audio source sparsity in the TF domain.

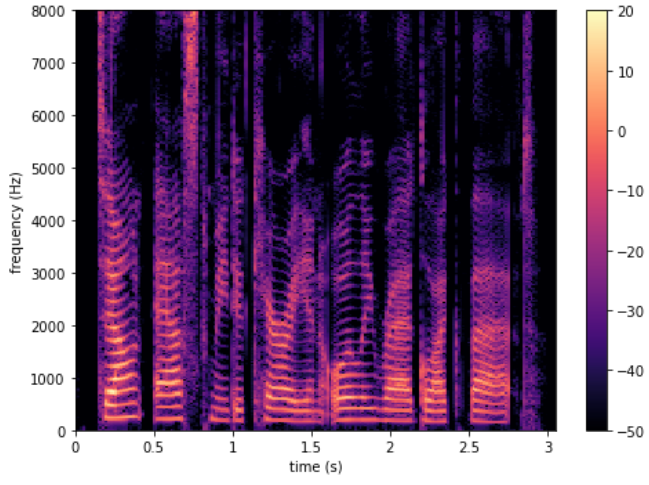
*Source 1*



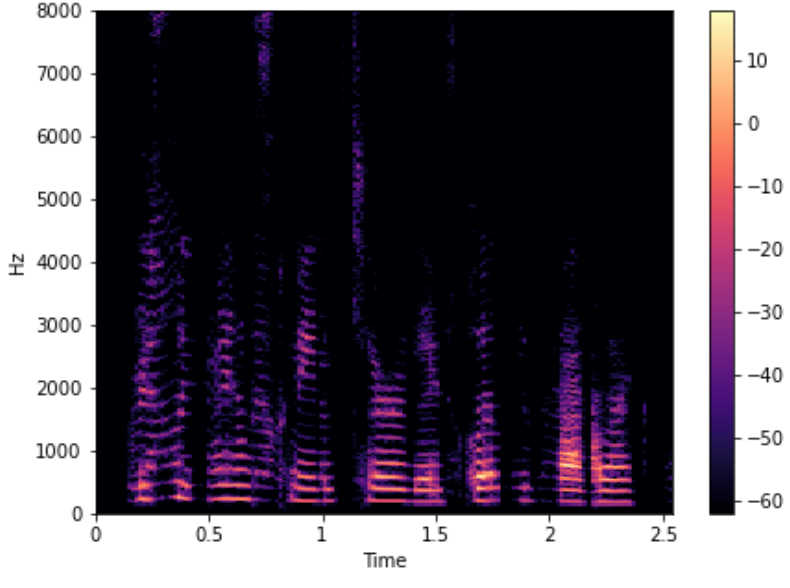

*Source 2*




*Source 3*




*Source 2 × Source 3: Most of time-frequency points indeed have low energy.*




*"All models are wrong, but some are useful."*

George Box (British statistician)

# Consequence 1: Unmixing with binary masking

W-disjoint orthogonality allows for separating the mixture into its component sources using binary masks:

$$\widehat{S}_j(k,m) = M_j(k,m)X_1(k,m),$$

where the mask is defined by:

$$M_j(k,m) = \begin{cases} 1 & \text{if } \mathcal{I}(k,m) = j \\ 0 & \text{otherwise} \end{cases}.$$

**The problem now is to estimate which source is active at each TF point.**

# Consequence 2: From mixture signals to mixture parameters

Let us recall the mixture model under the W-disjoint orthogonality assumption:

$$\begin{bmatrix} X_1(k,m) \\ X_2(k,m) \end{bmatrix} = \begin{bmatrix} 1 \\ a_{\mathcal{I}(k,m)} \exp\left(-i2\pi \frac{k\delta_{\mathcal{I}(k,m)}}{N}\right) \end{bmatrix} S_{\mathcal{I}(k,m)}(k,m).$$

In this model, **the ratio of the microphone signals in the STFT domain does not depend on the source signal but only on the mixing parameters corresponding to the active source:**

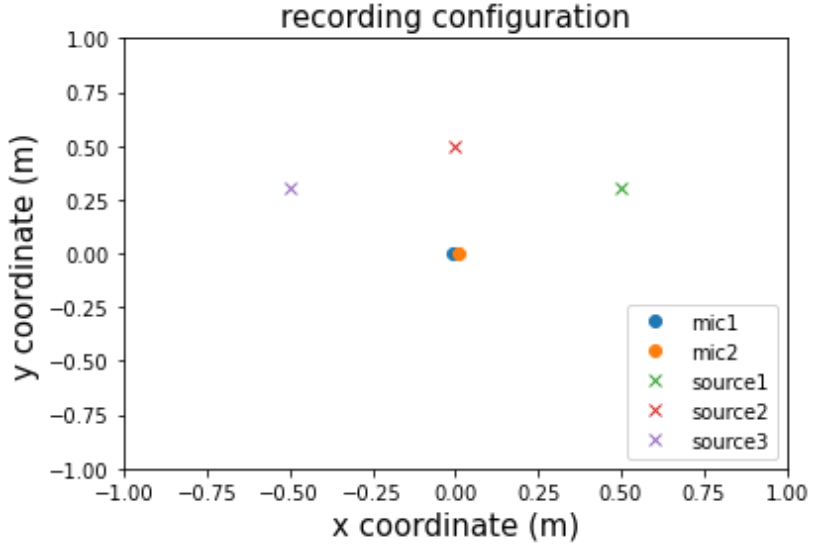
$$\frac{X_2(k,m)}{X_1(k,m)} = a_j \exp\left(-i2\pi \frac{k\delta_j}{N}\right), \qquad \forall (k,m) \in \Omega_j = \{(k,m) \mid \mathcal{I}(k,m) = j\}.$$

# Spatial diversity (assumption 2)

We recall that $a_j$ and $\delta_j$ encode the position of the $j$-th source relative to the microphones.

We assume that the **sources have different spatial locations**, that is

$$(a_j \neq a_l) \text{ or } (\delta_j \neq \delta_l), \qquad \forall j \neq l.$$



# 2D histogram of local attenuations and delays

- Let us define the set of local attenuations and delays by:

$$\widehat{a}(k,m) = \left| \frac{X_2(k,m)}{X_1(k,m)} \right|, \quad \widehat{\delta}(k,m) = -\frac{1}{2\pi k/N} \arg\left( \frac{X_2(k,m)}{X_1(k,m)} \right), \qquad \forall k,m.$$

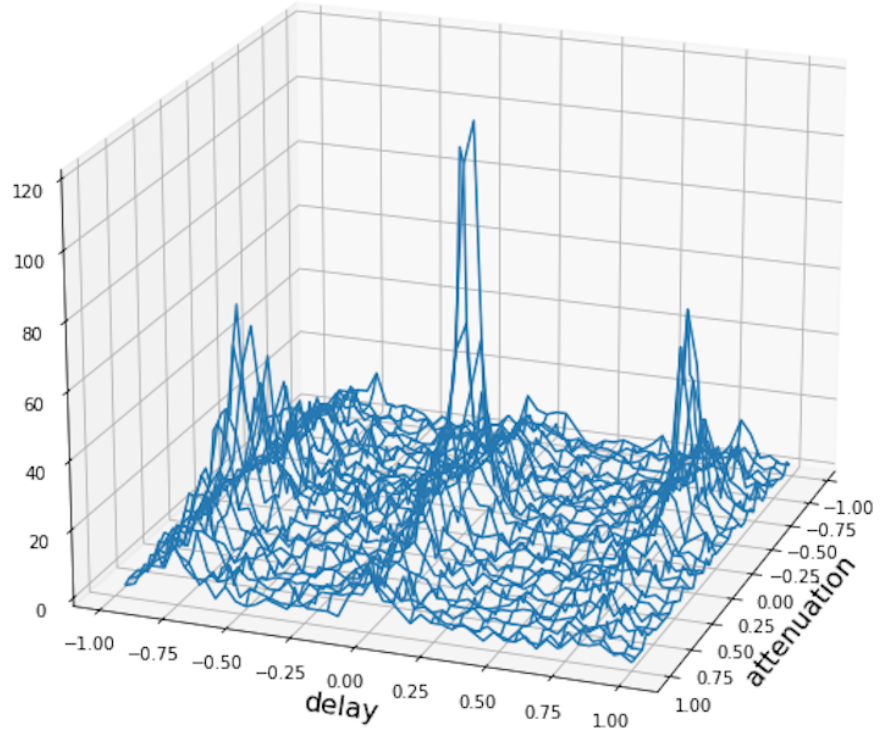
- According to the anechoic mixture model and WDO assumption, we should have:

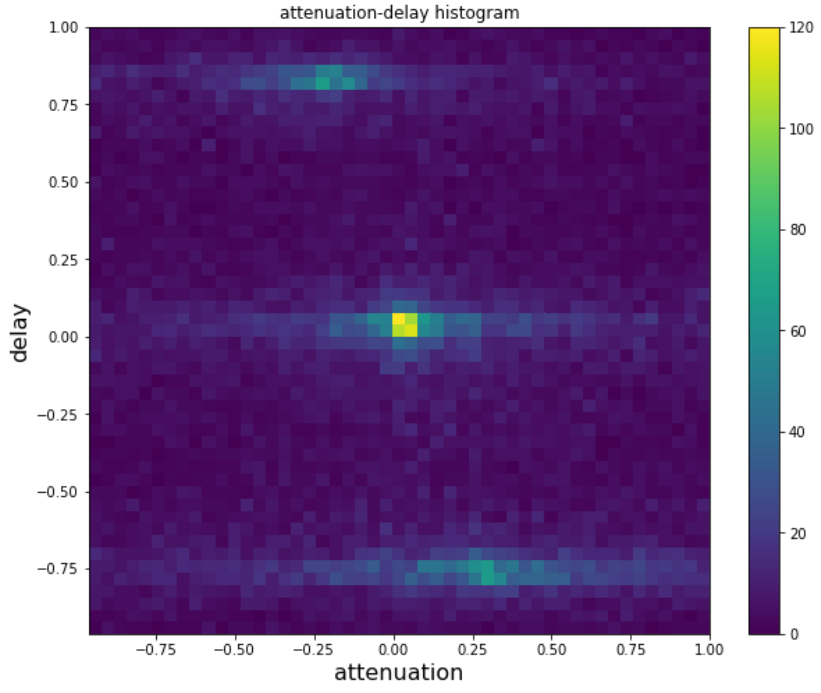
$$\widehat{a}(k,m) = a_j, \quad \widehat{\delta}(k,m) = \delta_j, \qquad \forall (k,m) \in \Omega_j = \{(k,m), \mathcal{I}(k,m) = j\}.$$

The local attenuations and delays that are computed from the mixture signals can thus only take values among the actual mixing parameters that are assumed to be different for different sources.

The main principle of DUET is thus to build a 2D histogram of the (observed) local attenuations and delays and to detect the peaks in this histogram. These peaks are expected to correspond to the "true" values of the mixing parameters.

# 2D histogram of local attenuations and delays: Example




In practice, because not all the assumptions are strictly satisfied, the local attenuations and delays are not precisely equal to the mixing parameters, but they will cluster around them. We can use peak picking methods to estimate them. We need a metric to measure the proximity.

# The DUET algorithm

1. Compute the STFT of both mixture signals $X_1(k,m)$ and $X_2(k,m)$.

2. Compute the ratio of the two mixtures and extract the local attenuations and delays:

$$\widehat{a}(k,m) = \left| \frac{X_2(k,m)}{X_1(k,m)} \right|, \quad \widehat{\delta}(k,m) = -\frac{1}{2\pi k/N} \arg\left( \frac{X_2(k,m)}{X_1(k,m)} \right), \qquad \forall k,m.$$

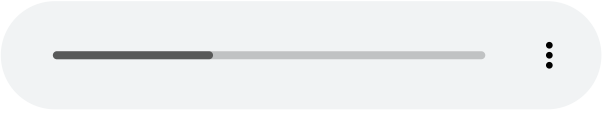
3. Compute a 2D histogram of the local attenuations and delays and estimate the mixing parameters $\{(\widehat{a}_j, \widehat{\delta}_j)\}_{j=1}^{J}$ by peak picking.

4. Compute the binary masks:

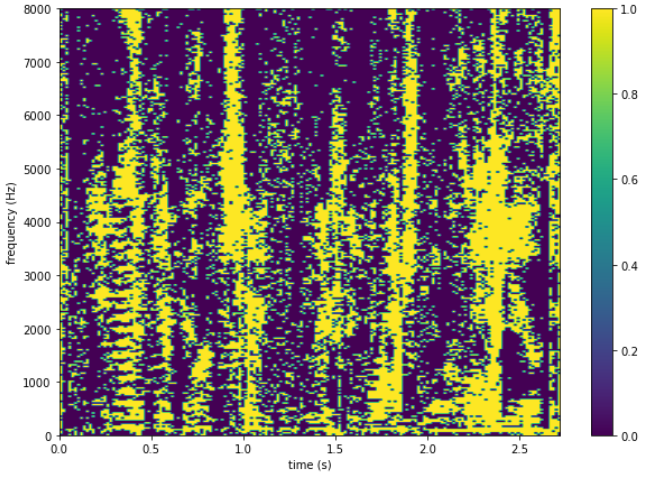
$$M_j(k,m) = \begin{cases} 1 & \text{if } (\widehat{a}(k,m), \widehat{\delta}(k,m)) \approx (\widehat{a}_j, \widehat{\delta}_j) \\ 0 & \text{otherwise} \end{cases}.$$

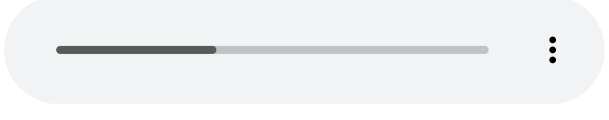
5. Estimate the sources by $\widehat{S}_j(k,m) = M_j(k,m) X_1(k,m)$.

6. Compute the inverse STFT of $\widehat{S}_j(k,m)$ to get the estimated time-domain source signals $\widehat{s}_j(n)$ .

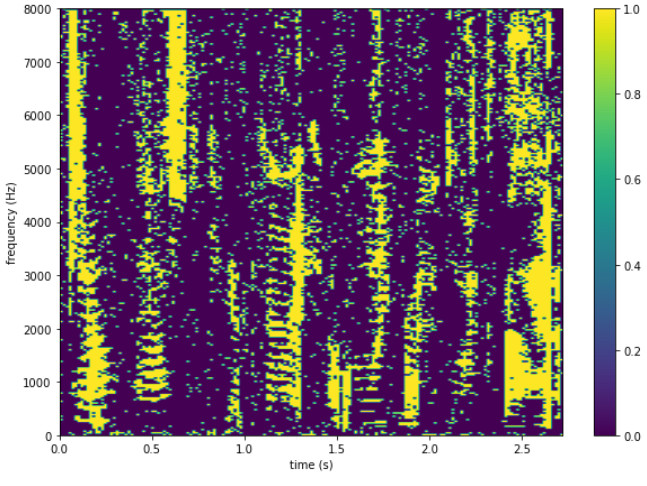


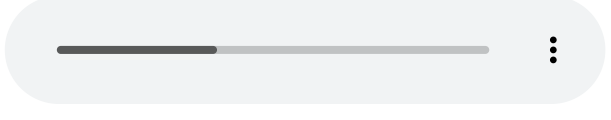
## Mask 1



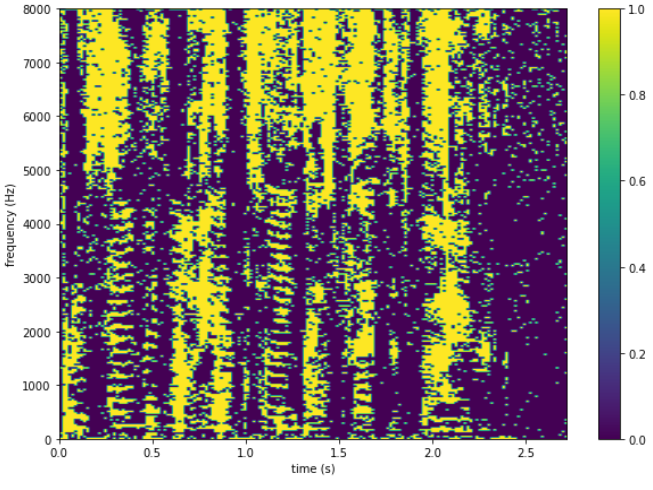



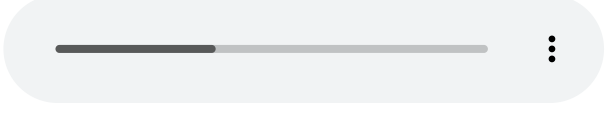
## Mask 2



## Mask 3



# Lab Work: Implementation and evaluation of DUET

- 4h Lab Work placed at the end of the course (this year, October 25-th)
- Todo list:
  - Implement DUET (from scratch...?) in Matlab or in Python
  - Simulate anechoic mixtures of speech signals
  - Evaluate the separation performance (with the output SDR metric, averaged over sources)
  - Extend the study to reverberant mixtures
    - Use existing RIR simulators and generate simulated image signals
    - Evaluate the separation performance as a function of the amount of reverberation (reverberation time and/or DRR).
    - If time allows, run DUET on real speech mixtures and listen to the separated signals

E. Habets, Room impulse response generator, Technical Report, 2006. C code available at https://www.audiolabs-erlangen.de/fau/professor/habets/software/rir-generator.

R. Scheibler, E. Bezzam & I. Dokmanić, Pyroomacoustics: A Python package for audio room simulations and array processing algorithms, IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 2018. Python code available at https://pypi.org/project/pyroomacoustics/.

# Source separation in more realistic acoustic conditions

## Generalization of DUET and methods based on statistical signal processing

S. Makino, T. W. Lee & H. Sawada (Eds.), Blind speech separation, Springer, 2007.

E. Vincent et al., Probabilistic modeling paradigms for audio source separation, in Machine Audition: Principles, Algorithms and Systems, IGI global, 2011.

S. Gannot et al., A consolidated perspective on multimicrophone speech enhancement and source separation, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2017.

E. Vincent, T. Virtanen & S. Gannot (Eds.), Audio source separation and speech enhancement, John Wiley & Sons, 2018.

# General principle

- Let us consider a multichannel recording set-up with $J$ sound sources and $I$ microphones, in a "natural indoor" environment.
- Here, the anechoic model is oversimplistic, since there are reverberations.
- We thus define a parametric convolutive mixture model, depending on source-to-microphone RIRs.
- We then express the mixture model in the STFT domain. This leads to some matrix system with source signals as inputs and observed mixture/microphone signals as output.
- We need to (a) estimate the parameters of this system (similarly to what we did for the attenuation factors and delays in DUET), and (b) recover the source signals.
- Methods differ depending on if $I \geq J$ (determined and over-determined setup) or $I < J$ (under-determined set-up). The latter case is more difficult, since we cannot directly invert the system.

# Mixture model (in the time domain)

- Each source signal $s_j$ is convolved with a multichannel RIR $a_{i,j}$ modeling its propagation to microphone $i$, resulting in the image source signal $\tilde{s}_{i,j}$:

$$\tilde{s}_{i,j}(n) = a_{i,j}(n) \star s_j(n) = \sum_{p=0}^{P-1} a_{i,j}(p) s_j(n-p),$$

  where $P$ is the effective length of the RIR (assumed identical for all RIRs).

- For each microphone, the microphone signal (or mixture signal) $x_i$ is the sum of the corresponding image source signals $\tilde{s}_{i,j}$:

$$x_i(n) = \sum_{j=1}^{J} \tilde{s}_{i,j}(n) = \sum_{j=1}^{J} \sum_{p=0}^{P-1} a_{i,j}(p) s_j(n-p).$$

- In this context, the RIRs $a_{i,j}$ are often referred to as the mixing filters.

# Mixture model (in the transformed domain)

- The general idea is to process the separation in the Fourier transformed domain, where the convolution operation transforms into a product at each frequency bin, which is much easier to handle. However, unfortunately, things are not so simple!

- Reminder: Under certain conditions, we have:

$$a_{i,j}(n) \star s_j(n) \overset{\text{DFT}}{\longleftrightarrow} A_{N,i,j}(k) S_{N,j}(k).$$

  Typically, the length $N$ of the DFT must be larger than the length of the convolution $P + N_s - 1$, where $N_s$ is the length of $s_j$.

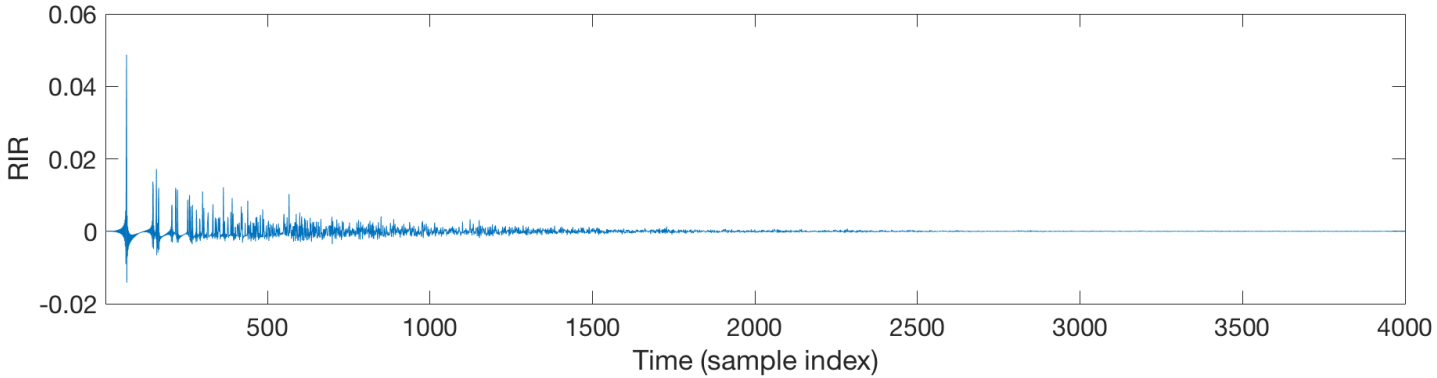
- This is NOT verified for the STFT !!!

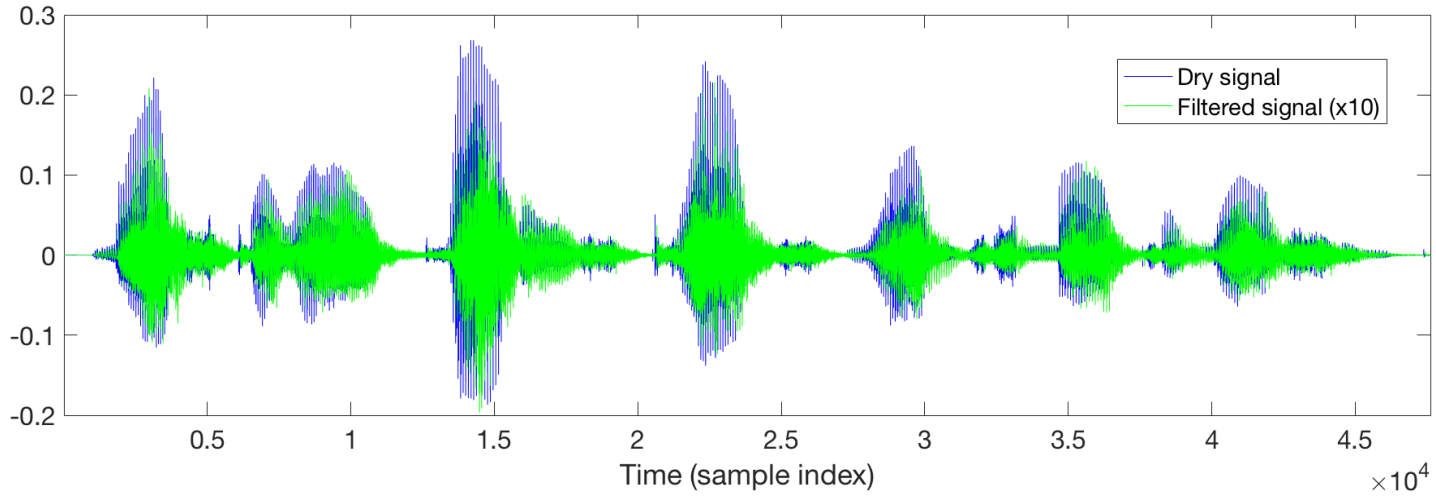
  In the STFT, we apply the DFT on short-term frames, which are... short! In practice, $N$ is often (much) shorter than $P$.

- In summary: $\text{STFT}[a_{i,j}(n) \star s_j(n)] = \tilde{S}_{i,j}(k,m) \neq A_{N,i,j}(k) S_j(k,m).$

# Illustration: A speech signal filtered with a simulated RIR

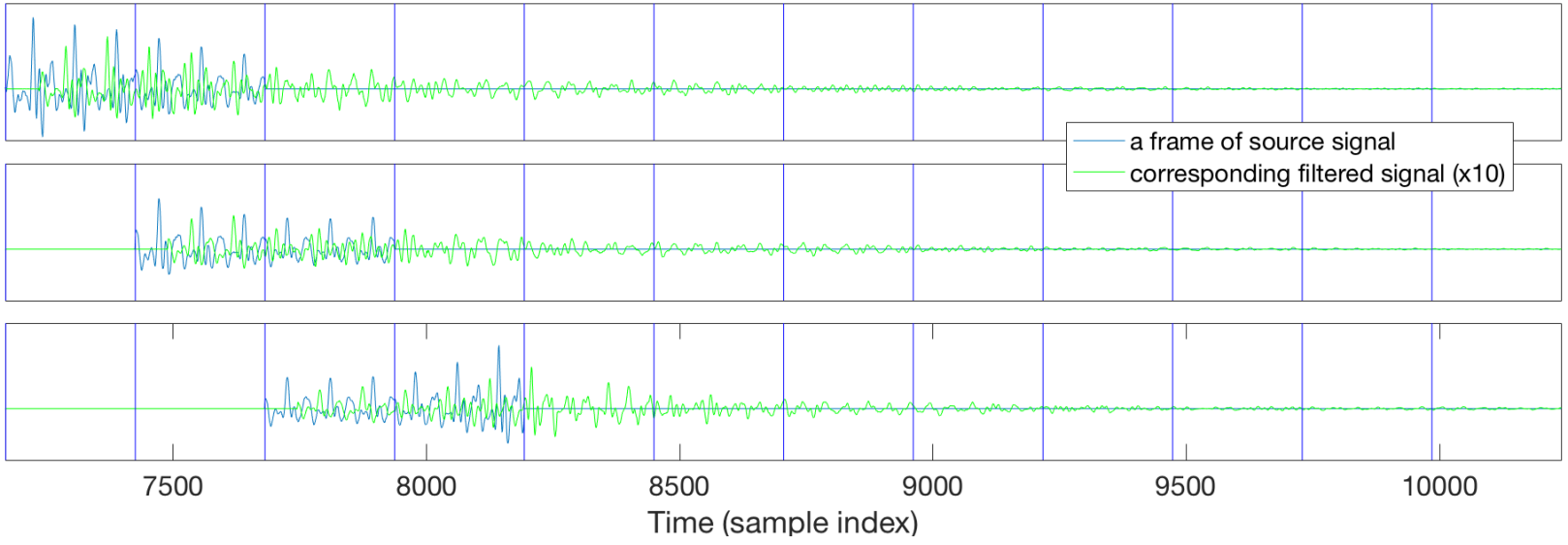
RIR generator from Audio Labs Erlangen, $T_{60} = 300\text{ms}$, $F_s = 16000\text{kHz}$.




E. Habets, Room impulse response generator, Technical Report, 2006. Software available at https://www.audiolabs-erlangen.de/fau/professor/habets/software/rir-generator.

# A closer look




- $N = 512$, $H = \frac{N}{2}$; blue vertical lines mark the frame boundaries at multiples of $H$.
- The convolution applied to an STFT frame of signal has notable effects in the future frames.
- The effect of signal filtering on a given frame is a combination (summation) of the effect of filtering that frame and the effect of filtering the previous frames.
- Therefore: $\tilde{S}_{i,j}(k,m) \neq A_{N,i,j}(k) S_j(k,m)$.

# The narrowband approximation

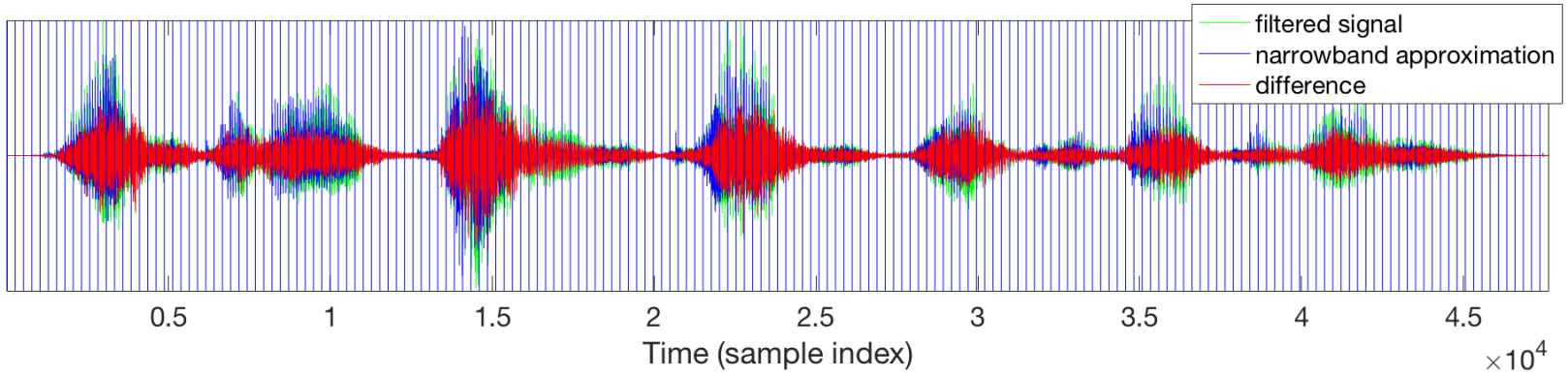
The vast majority of audio source separation methods adopts the following narrowband approximation (a.k.a the multiplicative transfer function model):

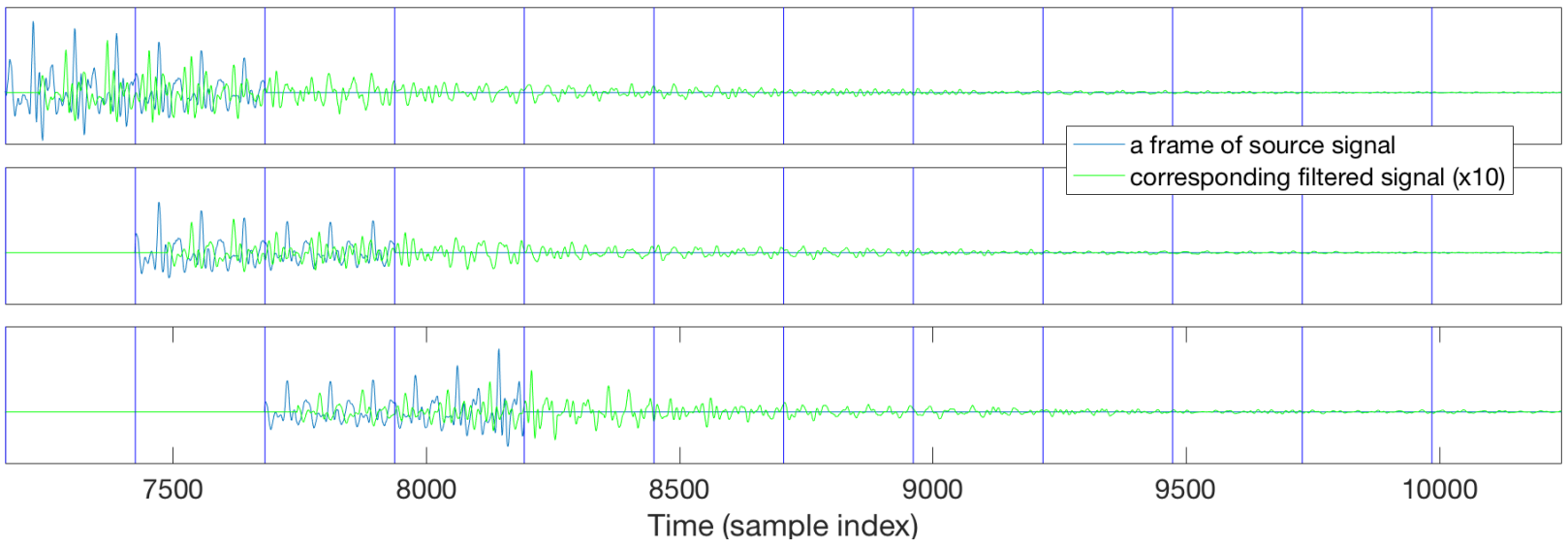
$$\tilde{S}_{i,j}(k,m) \approx A_{N,i,j}(k) S_j(k,m).$$

- This is true not only for source separation, but for a lot of methods dealing with STFT-domain processing (e.g., speech enhancement, sound source localization, etc).

# Influence of $N$

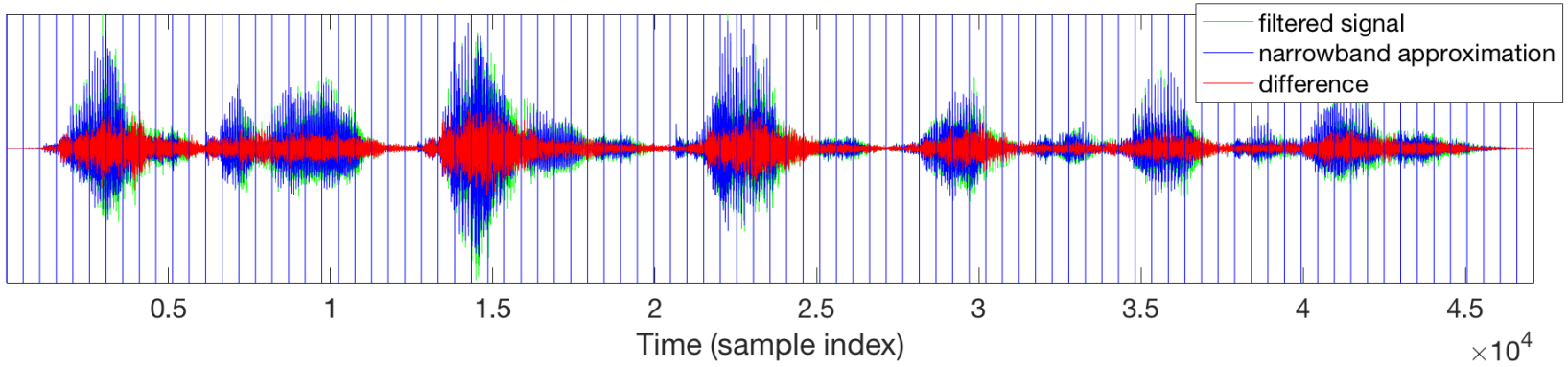
- $N = 512,\ \mathrm{SDR} = 1.8\mathrm{dB}$ (about 66% error!)

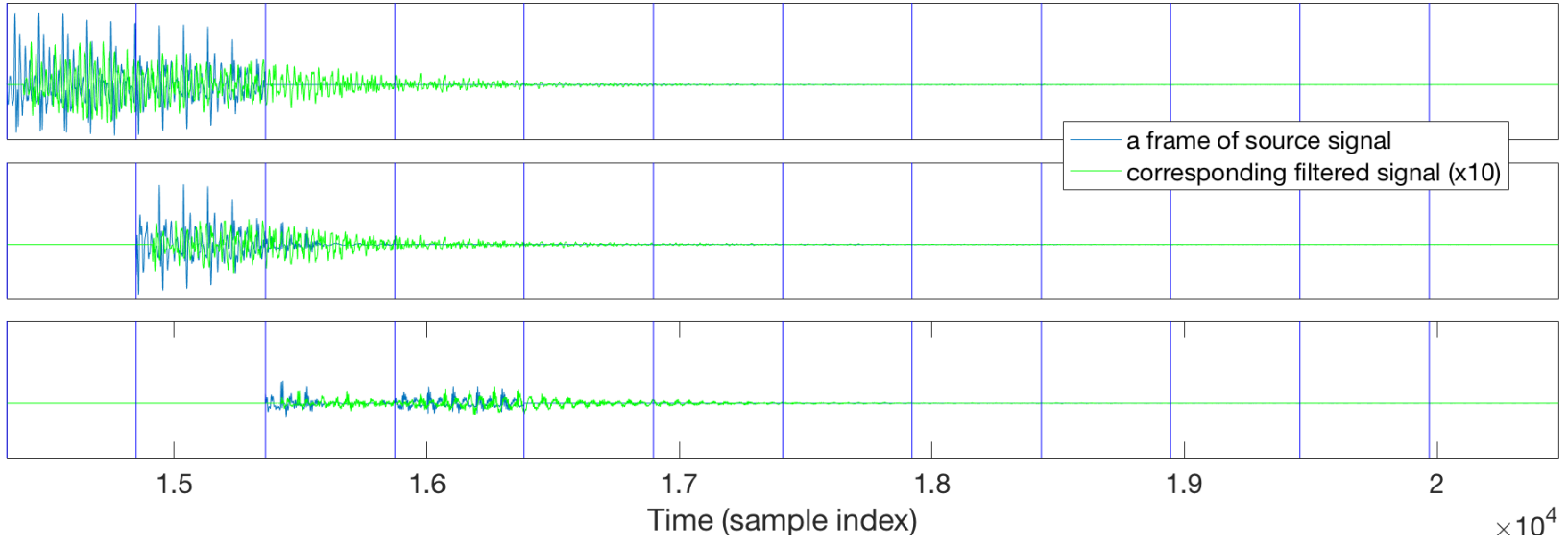


# Influence of $N$

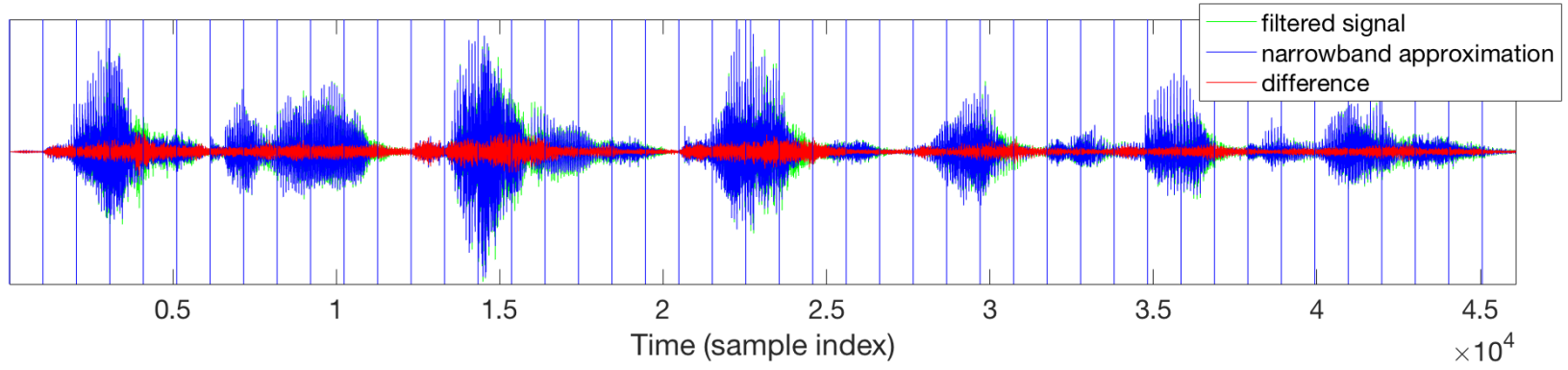
- $N = 1024,\ \mathrm{SDR} = 3.8\mathrm{dB}$

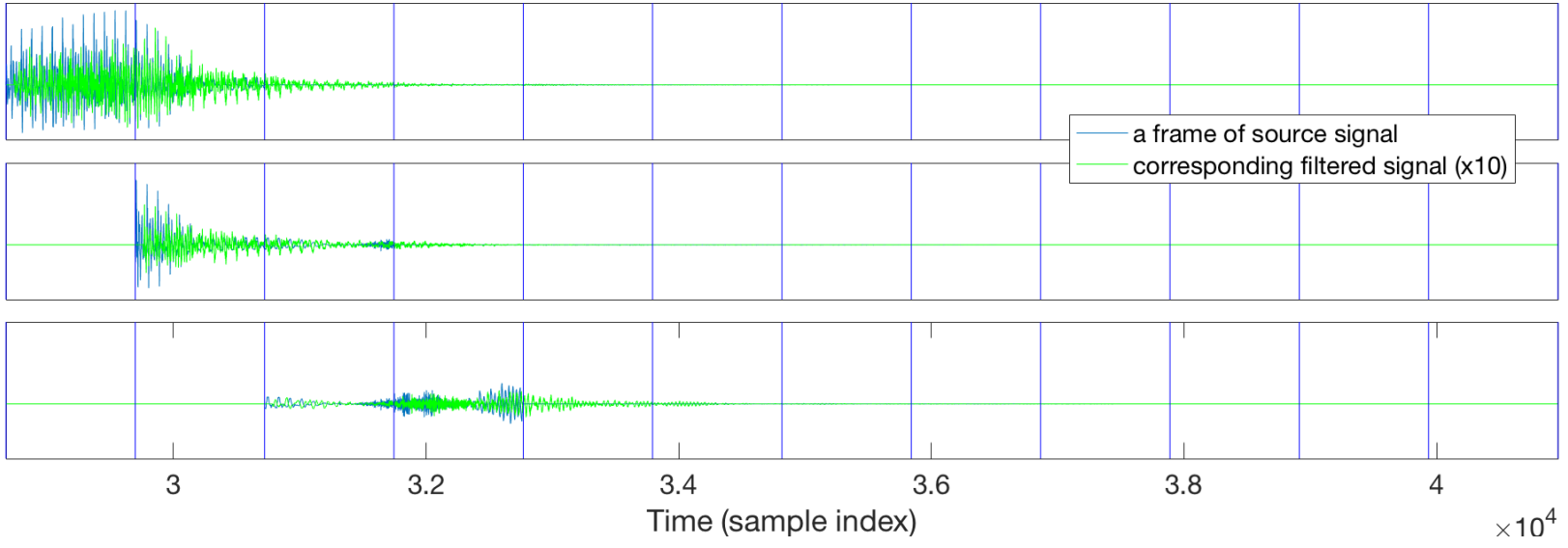


# Influence of $N$

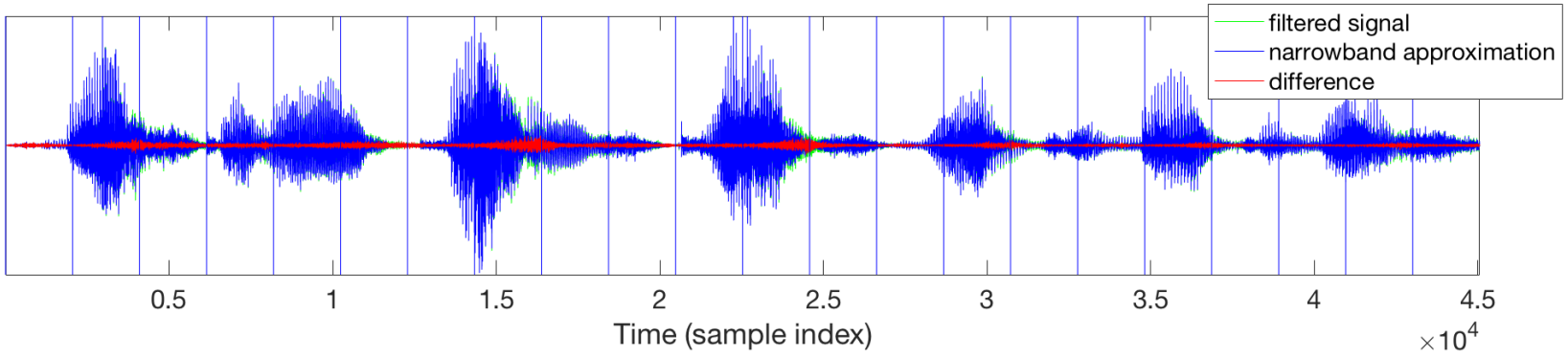
- $N = 2048,\ \mathrm{SDR} = 6.6\mathrm{dB}$

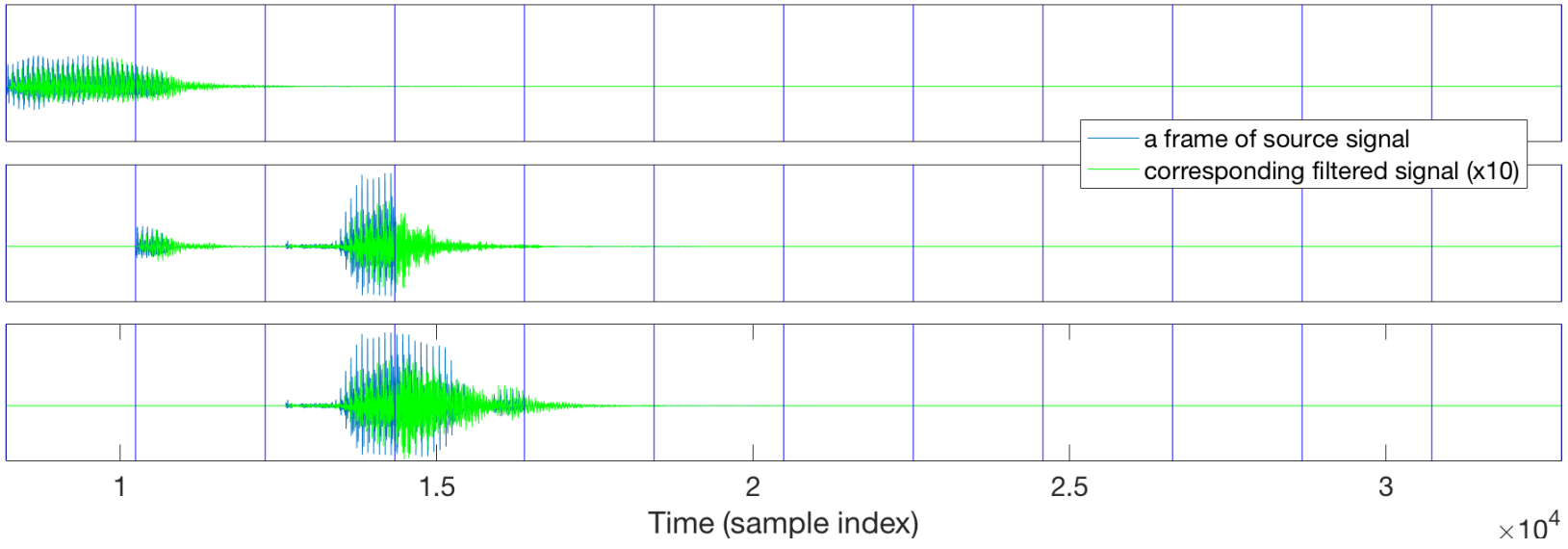


# Influence of $N$

- $N = 4096,\ \mathrm{SDR} = 10.5\mathrm{dB}$



# Influence of $N$

- In practice, we generally cannot use too large values of $N$ because:
  - The WDO assumption will fail if $N$ is large. The larger $N$, the higher the chance that the source spectrum sparsity decreases and several sources overlap.
  - Some methods require the signal local stationarity assumption (they use short-term parameters such as source signal PSD, which have sense only for short-term frames due to the non-stationarity of audio/speech signals).

The narrowband assumption is widely used for the "small" values of $N$ generally used in speech/audio analysis (e.g., $512$ or $1024$ for $16$-$\mathrm{kHz}$ speech signals), even if the amount of error is large.

- Remember: *"All models are wrong, but some are useful."* This one is VERY useful!

# Mixture model (in the STFT domain)

- Reminder: We have: $x_i(n) = \sum_{j=1}^{J} a_{i,j}(n) \star s_j(n)$
- Using the narrowband approximation, the mixture becomes in the STFT domain, $\forall k \in \{0, ..., \frac{N}{2}\}$:

$$\begin{bmatrix} X_1(k,m) \\ X_2(k,m) \\ ... \\ X_I(k,m) \end{bmatrix} \approx \begin{bmatrix} A_{N,1,1}(k) & A_{N,1,2}(k) & ... & A_{N,1,J}(k) \\ A_{N,2,1}(k) & A_{N,2,2}(k) & ... & A_{N,2,J}(k) \\ ... & ... & ... & ... \\ A_{N,I,1}(k) & A_{N,I,2}(k) & ... & A_{N,I,J}(k) \end{bmatrix} \begin{bmatrix} S_1(k,m) \\ S_2(k,m) \\ ... \\ S_J(k,m) \end{bmatrix}.$$

- This system can be rewritten: $\mathbf{x}(k,m) = \mathbf{A}(k)\mathbf{s}(k,m)$.
- We need to (a) estimate the mixing filters $\mathbf{a}_j(k) = \left[A_{N,1,j}(k), A_{N,2,j}(k), ..., A_{N,I,j}(k)\right]^\top$, and (b) estimate the source vector $\mathbf{s}(k,m) = \left[S_1(k,m), ..., S_J(k,m)\right]^\top$.
- Without additional hypotheses/constraints, this is a very difficult task!

# A method based on the WDO assumption

- With the WDO assumption, the mixture becomes:

$$\begin{bmatrix} X_1(k,m) \\ X_2(k,m) \\ \dots \\ X_I(k,m) \end{bmatrix} \approx \begin{bmatrix} A_{N,1,\mathcal{I}(k,m)}(k) \\ A_{N,2,\mathcal{I}(k,m)}(k) \\ \dots \\ A_{N,I,\mathcal{I}(k,m)}(k) \end{bmatrix} S_{\mathcal{I}(k,m)}(k,m) = \mathbf{a}_{\mathcal{I}(k,m)}(k) S_{\mathcal{I}(k,m)}(k,m),$$

  where $\mathcal{I}(k,m) \in \{1, 2, ..., J\}$ indicates which source is active at TF point $(k,m)$.

- Let us transform $\mathbf{x}(k,m)$ so that it does not depend on the source signal anymore, but only on a "spatial" vector, e.g.:

  - $\overline{\mathbf{x}}(k,m) = \frac{\mathbf{x}(k,m)}{\|\mathbf{x}(k,m)\|} \approx \frac{\mathbf{a}_{\mathcal{I}(k,m)}(k)}{\|\mathbf{a}_{\mathcal{I}(k,m)}(k)\|} =$ normalized mixing filter vector
  - $\overline{\mathbf{x}}(k,m) = \frac{\mathbf{x}(k,m)}{X_1(k,m)} \approx \frac{\mathbf{a}_{\mathcal{I}(k,m)}(k)}{A_{N,1,\mathcal{I}(k,m)}(k)} =$ relative transfer function (RTF) vector)

- Let us denote by $\overline{\mathbf{a}}_{\mathcal{I}(k,m)}(k)$ this "spatial" vector.

S. Araki et al., Underdetermined blind sparse source separation for arbitrarily arranged multiple sensors. Signal processing, 2007.

# Generalization of DUET

- Similarly to DUET, we have to:
  - (a) identify the vectors $\{\overline{\mathbf{a}}_j(k)\}_{j=1}^J$ (see next slide),
  - (b) associate each TF bin $(k,m)$ to the predominant source using a distance $d$ between $\overline{\mathbf{x}}(k,m)$ and each $\overline{\mathbf{a}}_j(k)$:

$$\mathcal{I}(k,m) = \arg\min_j d\big(\overline{\mathbf{x}}(k,m), \overline{\mathbf{a}}_j(k)\big).$$

  - (c) compute the binary masks:

$$M_j(k,m) = \begin{cases} 1 & \text{if } j = \mathcal{I}(k,m) \\ 0 & \text{otherwise} \end{cases}.$$

  - (d) estimate the (image) source signals by $\widehat{\widetilde{S}}_j(k,m) = M_j(k,m) X_1(k,m)$.

    (and compute the inverse STFT to have the time-domain signal).

# Identifying the directional vectors

- To identify $\{\overline{\mathbf{a}}_j(k)\}_{j=1}^J$, we can use the $k$-means algorithm, or any similar vector clustering algorithm, applied on the dataset $\{\overline{\mathbf{x}}(k,m)\}_{m=1}^M$:
  - Random (or smarter) initialisation of $\{\overline{\mathbf{a}}_j(k)\}_{j=1}^J$
  - Iterate untill convergence:
    - $\forall m \in \{1,...,M\}$, compute $\mathcal{I}(k,m) = \arg\min_j d\big(\overline{\mathbf{x}}(k,m), \overline{\mathbf{a}}_j(k)\big)$.
    - $\forall j \in \{1,...,J\}$, update $\overline{\mathbf{a}}_j(k) = \text{mean}[\overline{\mathbf{x}}(k,m) \mid \mathcal{I}(k,m) = j]$
- Note: We have one vector clustering procedure for each frequency bin $k \in \{0,...,N/2\}$, as opposed to a unique 2D histogram in DUET.

  Here, we must have $M \gg J$ for the vector clustering to be efficient.

# Statistical signal processing methods for audio source separation

- A general family of methods based on statistical models of source signals and Bayesian framework.
- Very nice features:
  - Does not need the WDO assumption.
  - Allows for the presence of noise (background diffuse noise + sensor noise, denoted $u_i(n)$).
    We have here: $x_i(n) = \sum_{j=1}^{J} a_{i,j}(n) \star s_j(n) + u_i(n), \quad \forall i \in \{1, ..., I\}.$
  - Works for underdetermined mixtures ($I < J$).

E. Vincent et al., Probabilistic modeling paradigms for audio source separation, in Machine Audition: Principles, Algorithms and Systems, IGI global, 2011.

A. Ozerov & C. Févotte, Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation, IEEE Transactions on Audio, Speech, and Language Processing, 2010.

N. Duong, E. Vincent & R. Gribonval, Under-determined reverberant audio source separation using a full-rank spatial covariance model, IEEE Transactions on Audio, Speech, and Language Processing, 2010.

# Mixture model in the STFT domain: Convolutive + noise

- In the STFT domain, the mixture is here given by:

$$\begin{bmatrix} X_1(k,m) \\ X_2(k,m) \\ \dots \\ X_I(k,m) \end{bmatrix} \approx \begin{bmatrix} A_{N,1,1}(k) & A_{N,1,2}(k) & \dots & A_{N,1,J}(k) \\ A_{N,2,1}(k) & A_{N,2,2}(k) & \dots & A_{N,2,J}(k) \\ \dots & \dots & \dots & \dots \\ A_{N,I,1}(k) & A_{N,I,2}(k) & \dots & A_{N,I,J}(k) \end{bmatrix} \begin{bmatrix} S_1(k,m) \\ S_2(k,m) \\ \dots \\ S_J(k,m) \end{bmatrix} + \begin{bmatrix} U_1(k,m) \\ U_2(k,m) \\ \dots \\ U_I(k,m) \end{bmatrix},$$

  which can be rewritten as:

$$\mathbf{x}(k,m) \approx \sum_{j=1}^{J} \mathbf{a}_j(k) S_j(k,m) + \mathbf{u}(k,m).$$

# Spatial covariance matrix

- Assuming that the source signals are centered, let us define the spatial covariance matrix (SCM) of the (multichannel) microphone signal as:

$$\mathbf{\Sigma}_{\mathbf{x}}(k,m) = \mathbb{E}\left[\mathbf{x}(k,m)\mathbf{x}(k,m)^H\right].$$

  This matrix encodes the correlations across the different channels of the recorded signal due to the spatial distribution of the image source signals.

- Assuming that the different sources are uncorrelated, i.e., $\mathbb{E}\left[S_j(k,m)S_l(k,m)^*\right] = 0, \forall j \neq l$, and that the sources and the noise are uncorrelated too, we have:

$$\mathbf{\Sigma}_{\mathbf{x}}(k,m) \approx \sum_{j=1}^{J} \mathbf{a}_j(k)\mathbf{a}_j(k)^H v_{s_j}(k,m) + \mathbf{\Sigma}_{\mathbf{u}}(k,m),$$

  where $v_{s_j}(k,m) = \mathbb{E}\left[|S_j(k,m)|^2\right]$ is the PSD of source $j$,

  $\mathbf{\Sigma}_{\mathbf{u}}(k,m) = \mathbb{E}\left[\mathbf{u}(k,m)\mathbf{u}(k,m)^H\right]$ is the SCM of the noise.

# Estimation of the mixing filters with EVD

- Let us assume for a moment that $I \geq J$, i.e., the determined or over-determined case.
- Assuming that the power of the source signals is larger than the power of the noise, the eigenvalue decomposition (EVD) of $\mathbf{\Sigma_x}(k, m)$ should provide a series of eigenvectors and eigenvalues such that:
  - the eigenvectors associated to the largest eigenvalues will correspond to $\mathbf{a}_j(k)$ and span the so-called signal subspace;
  - the eigenvectors associated to the small eigenvalues will correspond to the noise subspace.
- There exist a set of methods for EVD in the literature, we do not detail them here.

# SCM estimation

- In practice, we can have an estimate of $\mathbf{\Sigma_x}(k,m)$ with:

$$\widehat{\mathbf{\Sigma}}_{\mathbf{x}}(k,m) = \sum_{p=m-m_1}^{m+m_2} \mathbf{x}(k,p)\mathbf{x}(k,p)^H,$$

  where $m_1$ and $m_2$ are arbitrary positive integer values, such that the estimate uses frames "around" the $m$-th frame.

- EVD is applied on this matrix estimate.

- Note that the fact that we use several frames somehow contradicts the stationary assumption, i.e., $v_{s_j}(k,m)$ can vary in time. But this is not a major problem, since we assume here that $\mathbf{a}_j(k)$ is constant over time. For mobile sound source, the problem is much more difficult to solve, since it is difficult to have a reliable $\widehat{\mathbf{\Sigma}}_{\mathbf{x}}(k,m)$ estimate.

# Direct mixture inversion

- If the noise is low, we can obtain the source estimate by direct inversion of the mixture:

$$\hat{\mathbf{s}}(k,m) = \widehat{\mathbf{A}}(k)^{-1}\mathbf{x}(k,m) = \widehat{\mathbf{A}}(k)^{-1}\mathbf{A}(k)\mathbf{s}(k,m) + \widehat{\mathbf{A}}(k)^{-1}\mathbf{u}(k,m) \approx \mathbf{s}(k,m).$$

- However, (a) this requires $I \geq J$ (if $I > J$ we use the pseudo-inverse matrix instead of the inverse matrix), and (b) there is a risk that $\widehat{\mathbf{A}}(k)^{-1}$ actually amplify the noise.

- Therefore, direct inversion of the mixture system is poorly used in practice.

- Instead, there is a series of more principled methods that takes the noise into account and that apply to under-determined mixtures ($I < J$).

  These methods are powerful but they require further estimation of source and noise PSDs.

  Let us see an example with the multichannel Wiener filter.

# The multichannel Wiener filter: Introduction

- Part of probabilistic / Bayesian methods for audio source separation.
- General framework for defining probability density function (pdf) models of source signals, deriving parameter estimation (in particular $\mathbf{A}(k)$ and source/noise PSDs), e.g. in the Maximum Likelihood sense, and finally obtain source signal estimates.
- This class of methods works for $I < J$, i.e., independently of $\mathbf{A}(k)$ invertibility!

# Source model

- The source signal distribution in the STFT domain is modeled with a zero-mean complex proper Gaussian distribution:

$$S_j(k,m) \sim \mathcal{N}_c\big(0, v_{s_j}(k,m)\big) = \frac{1}{|\pi v_{s_j}(k,m)|} \exp\left(-\frac{|S_j(k,m)|^2}{v_{s_j}(k,m)}\right),$$

  with $v_{s_j}(k,m) = E[|S_j(k,m)|^2]$.

- This means that both the real and imaginary part of $S_j(k,m)$ follow a zero-mean Gaussian distribution with variance $v_{s_j}(k,m)/2$ and are independant.
  The phase $\angle S_j(k,m)$ is assumed to follow an uniform distribution in $[0, 2\pi[$.

- Initially proposed for speech enhancement in (Ephraim & Malah, 1984) and for source separation in (Févotte & Cardoso, 2005), and then became very popular in audio processing.

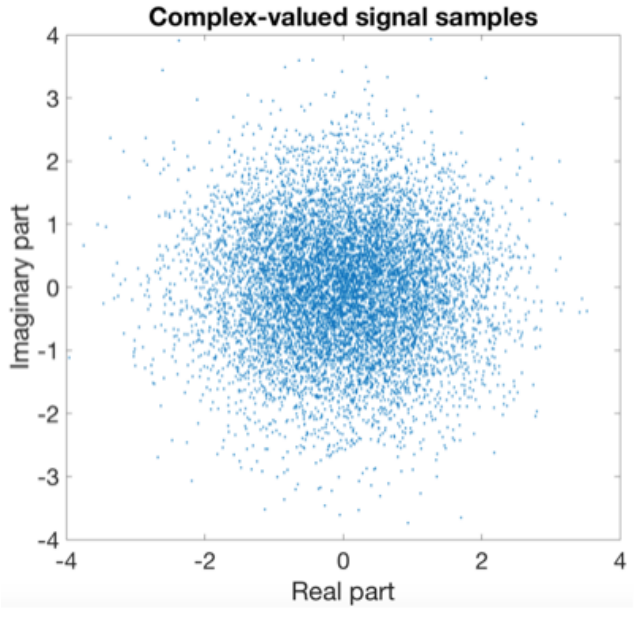
F. D. Neeser & J. L. Massey, Proper complex random processes with applications to information theory, IEEE Transactions on Information Theory, 1993.

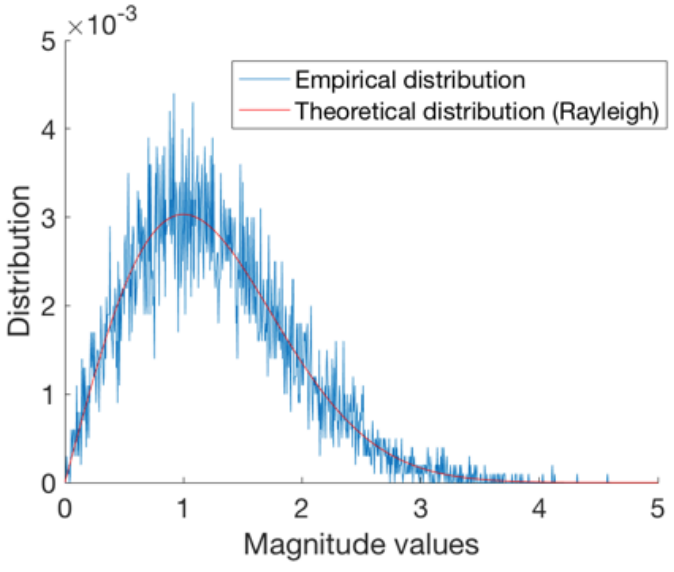
Y. Ephraim & D. Malah, Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984.

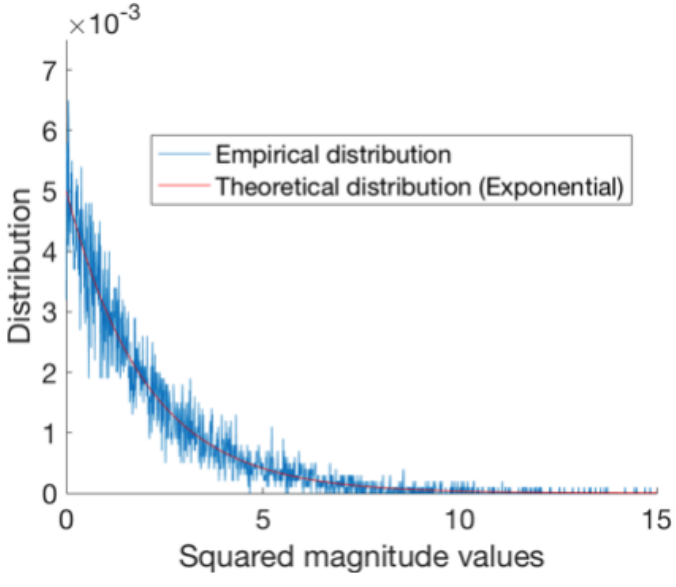
C. Févotte & J.-F. Cardoso, Maximum likelihood approach for blind audio source separation using time-frequency Gaussian source models, IEEE WASPAA, 2005.

A. Liutkus, B. Badeau & G. Richard, Gaussian processes for underdetermined source separation, IEEE Transactions on Signal Processing, 2011.

# Zero-mean complex proper Gaussian distribution



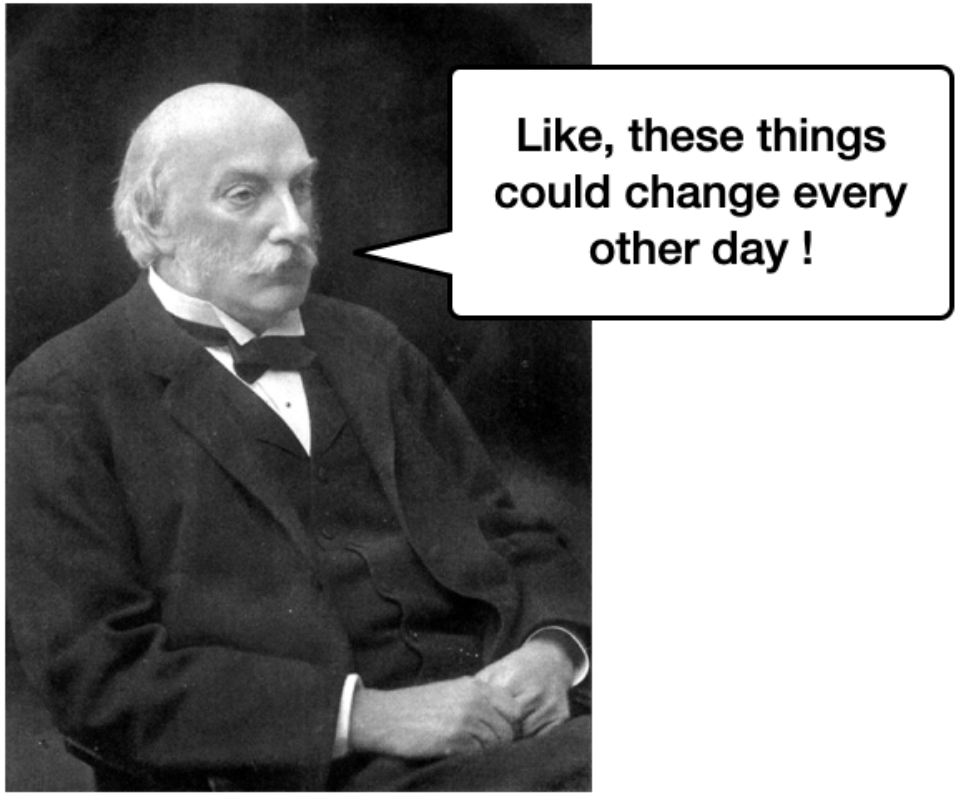




In this example, the variance $v_s = E[|S|^2]$ is equal to $2$.

# A bit of fun

The distribution of the modulus of the zero-mean complex proper Gaussian distribution is the Rayleigh distribution, named after Sir John William Strutt, baron of Rayleigh and thus better known as lord Rayleigh. Incidentally, this guy wrote: "Theory of sound" (two volumes, 1877-1878), regularly reissued...



# Source model

- The sources are assumed to be mutually independent:

$$\mathbf{s}(k,m) \sim \mathcal{N}_c\big(\mathbf{0}, \mathbf{\Sigma_s}(k,m)\big) = \frac{1}{|\pi \mathbf{\Sigma_s}(k,m)|} \exp\Big( - \mathbf{s}(k,m)^H \mathbf{\Sigma_s}(k,m)^{-1} \mathbf{s}(k,m)\Big)$$

  with the source covariance matrix $\mathbf{\Sigma_s}(k,m) = \mathbb{E}\left[\mathbf{s}(k,m)\mathbf{s}(k,m)^H\right] = \mathrm{diag}\big(\mathbf{v_s}(k,m)\big)$ ,
  with $\mathbf{v_s}(k,m) = \left[v_{s_j}(k,m), ..., v_{s_J}(k,m)\right]^\top$.

- The sources are also assumed to be individually independent across TF bins.

- NB: One parameter for each source and each TF bin !!! :-(

# Mixture model, posterior distribution, and source vector estimate

- For the noise, we assume: $\mathbf{u}(k,m) \sim \mathcal{N}_c\big(\mathbf{0}, \mathbf{\Sigma}_\mathbf{u}(k)\big)$, possibly with $\mathbf{\Sigma}_\mathbf{u}(k) = \mathrm{diag}\big(\mathbf{v}_\mathbf{u}(k)\big)$.
  Note: The noise is assumed stationary.

- It can be easily shown that: $\mathbf{x}(k,m) \sim \mathcal{N}_c\big(\mathbf{0}, \mathbf{\Sigma}_\mathbf{x}(k,m)\big)$,

  with $\mathbf{\Sigma}_\mathbf{x}(k,m) = \mathbf{A}(k)\mathbf{\Sigma}_\mathbf{s}(k,m)\mathbf{A}(k)^H + \mathbf{\Sigma}_\mathbf{u}(k)$.

- It can be (less easily!) shown that: $\mathbf{s}(k,m)|\mathbf{x}(k,m) \sim \mathcal{N}_c\big(\boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}}(k,m), \mathbf{\Sigma}_{\mathbf{s}|\mathbf{x}}(k,m)\big)$,

  with
  $$\boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}}(k,m) = \mathbf{\Sigma}_\mathbf{s}(k,m)\mathbf{A}(k)^H\mathbf{\Sigma}_\mathbf{x}(k,m)^{-1}\mathbf{x}(k,m)$$
  $$\mathbf{\Sigma}_{\mathbf{s}|\mathbf{x}}(k,m) = \mathbf{\Sigma}_\mathbf{s}(k,m) - \mathbf{\Sigma}_\mathbf{s}(k,m)\mathbf{A}(k)^H\mathbf{\Sigma}_\mathbf{x}(k,m)^{-1}\mathbf{A}(k)\mathbf{\Sigma}_\mathbf{s}(k,m).$$

The optimal estimate of $\mathbf{s}(k,m)$ in the MMSE sense is given by the mean vector of the posterior distribution, i.e.:

$$\hat{\mathbf{s}}(k,m) = \mathbb{E}\big[\mathbf{s}(k,m)|\mathbf{x}(k,m)\big] = \boldsymbol{\mu}_{\mathbf{s}|\mathbf{x}}(k,m).$$

# The multichannel Wiener filter

The solution can be rewritten:

$$\begin{aligned}\hat{\mathbf{s}}(k,m) &= \mathbf{\Sigma_s}(k,m)\mathbf{A}(k)^H\left(\mathbf{A}(k)\mathbf{\Sigma_s}(k,m)\mathbf{A}(k)^H + \mathbf{\Sigma_u}(k)\right)^{-1}\mathbf{x}(k,m)\\ &= \mathbf{W}(k,m)\mathbf{x}(k,m),\end{aligned}$$

where $\mathbf{W}(k,m) = \mathbf{\Sigma_s}(k,m)\mathbf{A}(k)^H\left(\mathbf{A}(k)\mathbf{\Sigma_s}(k,m)\mathbf{A}(k)^H + \mathbf{\Sigma_u}(k)\right)^{-1}$

is the multichannel Wiener filter (MWF).

- This is the generalization of the single-channel Wiener filter $W(k,m) = \frac{v_s(k,m)}{v_s(k,m)+v_u(k,m)}$ to the multichannel case.
- The MWF exploits the source spatial information encoded in $\mathbf{A}(k)$, the source spectral information encoded in $\mathbf{\Sigma_s}(k,m)$, and the noise characteristics encoded in $\mathbf{\Sigma_u}(k)$.

# The multichannel Wiener filter: Parameters estimation

- We need estimates of $\mathbf{A}(k)$, $\mathbf{\Sigma_s}(k,m)$, and $\mathbf{\Sigma_u}(k)$. Not an easy task! No general solution!
- A series of algorithms proposed in the literature, often based on the Expectation-Maximization (EM) algorithm, which general principle is:
  - Arbitrary initialization of the parameters. For example, if $I \geq J$ we can use EVD for initializing $\mathbf{A}(k)$.
  - Iterate between the following E-step and M-step:
    - E-step: The parameters being fixed to their current values, given the observed variables, estimate the latent (unobserved) variables (in the present case $\hat{\mathbf{s}}(k,m)$, estimated by Wiener filtering).
    - M-step: Given the observed variables and the current values of latent variables, update the parameter values.

The general principle is clear, but in practice EM algorithms for audio source separation can be complex. Moreover, there are difficulties that we have overlooked.

# Audio source separation in the STFT domain: difficulties

- The source permutation problem: $\mathbf{s}(k,m)$ and $\mathbf{A}(k)$ can be estimated only up to some source permutation, since a permutation on the columns of $\mathbf{A}(k)$ and the entries of $\mathbf{s}(k,m)$ doesn't affect the mixture. The separation is processed independently (in parallel) for each frequency bin $k$. Therefore, there can be source ordering permutation across different frequency bins. This can be solved by applying continuity contraints between reconstructed source signals and/or estimated filters at neighboring frequency bins, but this is a complex task.

- The gain ambiguity: $\mathbf{s}(k,m)$ and $\mathbf{A}(k)$ can be estimated only up to some scale factor, i.e., $\tilde{\mathbf{s}}(k,m) = \mathbf{A}(k)\mathbf{s}(k,m) = \mathbf{A}(k)/\alpha \times \alpha\mathbf{s}(k,m)$.

  - In some applications, one may interested in recovering the image source signals $\tilde{\mathbf{s}}(k,m)$.

  - We may separate the image source signals $\tilde{\mathbf{s}}(k,m)$ first, and then estimate the "dry" source signals $\mathbf{s}(k,m)$ by using dereverberation techniques (not presented here).

- Too many parameters: In the MWF, we have one source covariance matrix $\mathbf{\Sigma}_{\mathbf{s}}(k,m)$ to estimate for each TF bin! This is too many! (compared with the number of observations).

# Nonnegative matrix factorization

- A general factor analysis or dimension reduction technique for factorizing a nonnegative-definite matrix into a product of two (much) smaller nonnegative-definite matrices.
- Widely applied for almost two decades in audio processing, in particular for the modeling of power spectrograms (observed power spectrograms and PSDs). Here, we apply it to model the source PSD/variance parameters.
- Let $\mathbf{V}$ denote the $K \times M$ PSD matrix of a source signal $s(n)$ (i.e., $\mathbf{V}_{k,m} = v(k,m) = \mathbb{E}[|S(k,m)|^2]$) ($K = \frac{N}{2} + 1$). We set:

$$\mathbf{V} \approx \mathbf{W}\mathbf{H}$$

where $\mathbf{W}$ is a nonnegative $K \times D$ spectral pattern matrix,

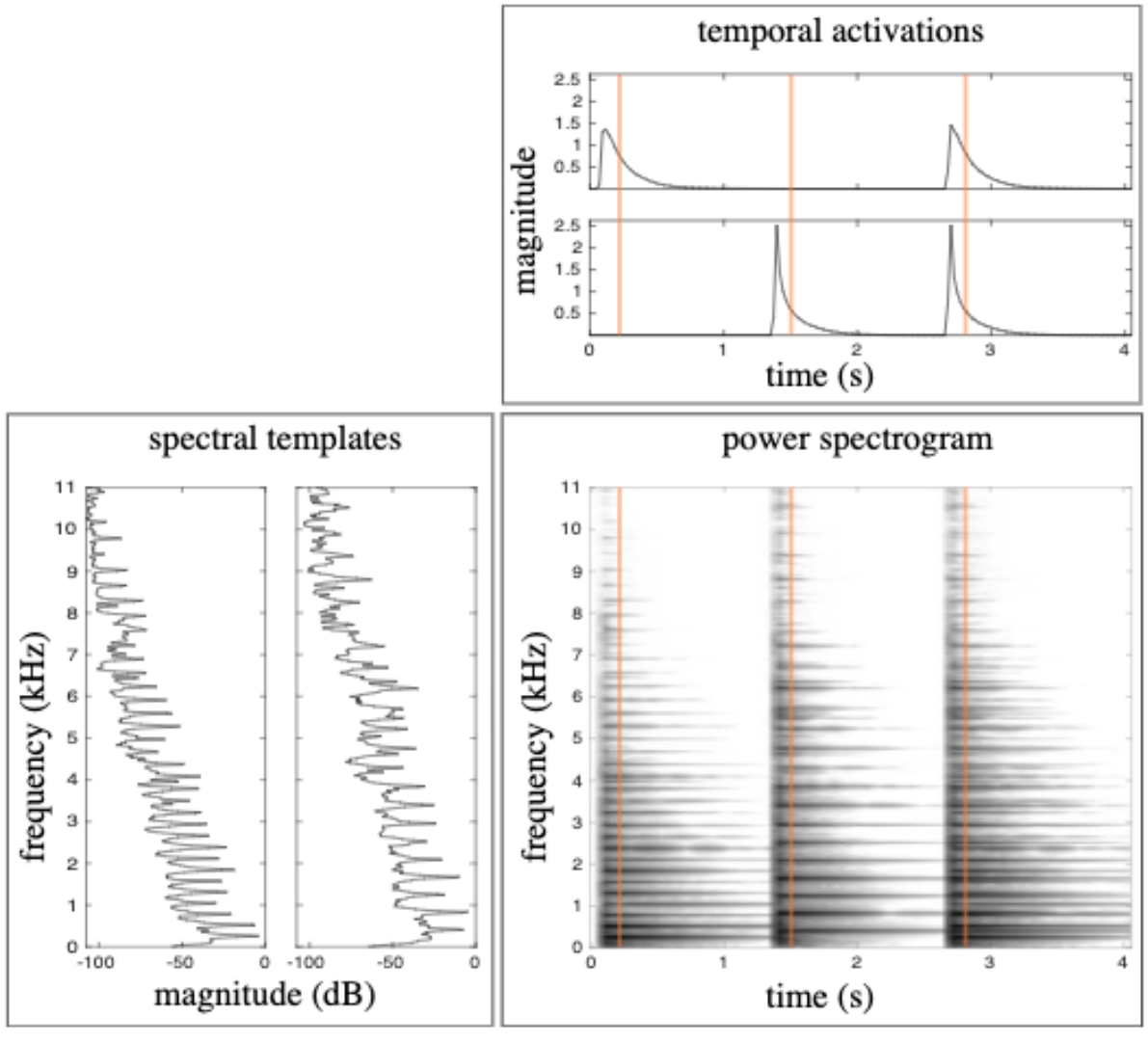
$\mathbf{H}$ is a nonnegative $D \times M$ temporal activation matrix,

$D$ is set such that $D(K+M) \ll KM$.

C. Févotte et al., Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis, Neural Computation, 2009.

# Example of NMF applied to audio

Two piano notes, first played successively and then played together.

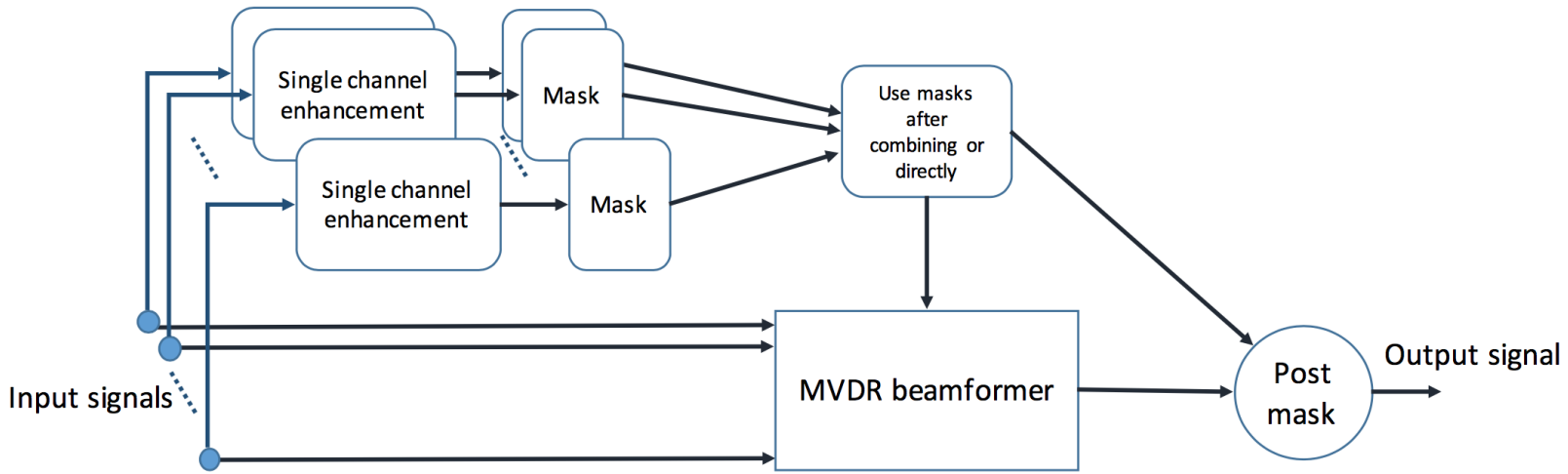


# NMF parameters estimation

- Given $\mathbf{V}$ and $D$, there exist an (iterative) optimization algorithm for estimating $\mathbf{W}$ and $\mathbf{H}$ (with different variants).
- In the present audio source separation framework, we have one matrix $\mathbf{V}_j$ for each source, and a set of NMF matrices $\{\mathbf{W}_j, \mathbf{H}_j\}_{j=1}^{J}$ to estimate.
- The estimation of these parameters is included in the EM algorithm.
- Typically $D$ is set to a few tens.
- The number of NMF source parameters $J \times D \times (K + M)$ is thus much lower than the initial number of source parameters $J \times K \times M$. This makes the whole EM algorithm more robust.

# As a conclusion: An example of combination of DL + spatial processing

- DNN-based single-channel speech enhancement (to extract the spectral information) and spatial filtering (to exploit the spatial info)
- DNNs provide masks, used to select speech/noise points, used to estimate speech/noise covariance matrices, used to build beamforming filters (= spatial filters).




H. Erdogan et al. Improved MVDR Beamforming Using Single-Channel Mask Prediction Networks, Proc. of Interspeech Conf., 2016.