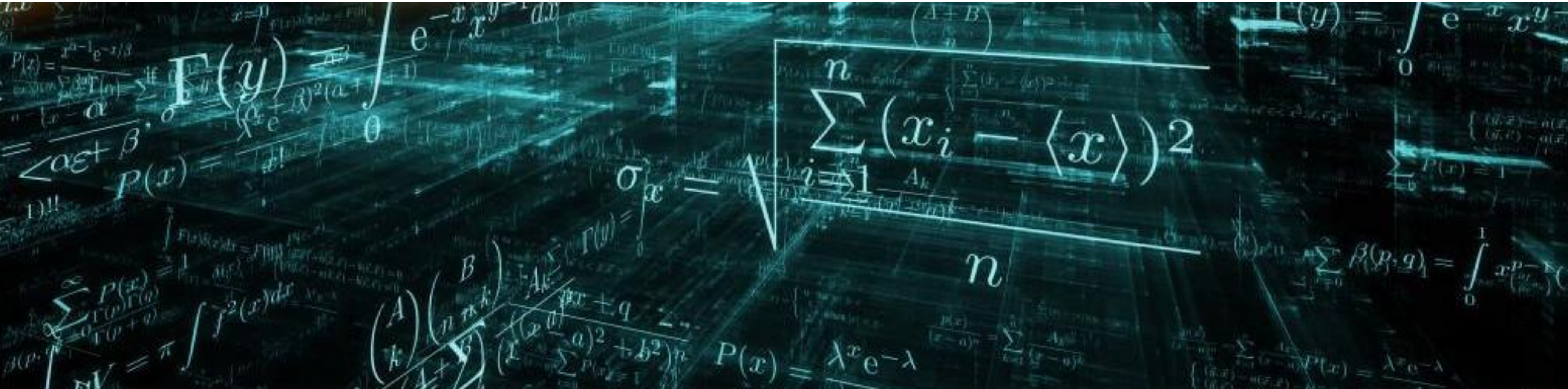
Universität Basel

Computer Science / 15731-01 / 2025

Multimedia Retrieval

Chapter 12: Acoustic Features

Dr. Roger Weber, roger.weber@gmail.com



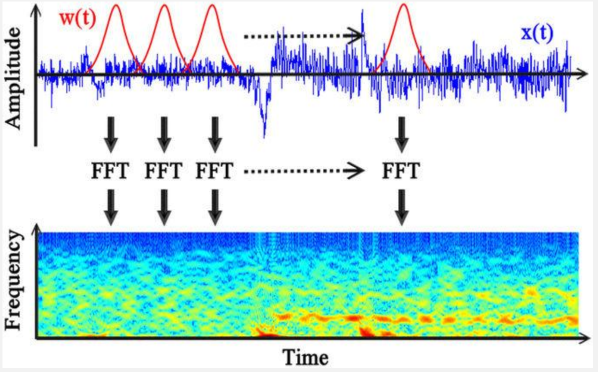
12.1 Introduction

- Sound has two complementary meanings. In physics it is a mechanical wave that travels through a medium. In perception it is the sensation those waves create when they reach a listener. Both perspectives describe the same phenomenon and are important for later audio analysis.
- **Physical Perspective:** Sound begins when something vibrates and disturbs the surrounding medium. For example, a vibrating loudspeaker membrane pushes air molecules back and forth, creating alternating regions of compression and rarefaction. These variations travel outward as a wave with a specific wavelength, frequency, pressure, speed and direction. The wave moves through the medium, but the particles themselves only oscillate around their resting positions. Because sound depends on interactions among particles, it cannot travel through empty space. The relationship between wavelength and frequency is set by the wave speed, and in air under standard conditions that speed depends on temperature. Sound travels faster in liquids and faster still in solids, where dense molecular structures pass vibrations more efficiently. As waves move they spread, bounce off surfaces, bend at boundaries between different materials and gradually lose energy. Studying these behaviors lets one trace how sound travels through an environment and how it interacts with objects before reaching the listener.
  - The human ear can hear sounds with frequencies between 20Hz and 20kHz. This corresponds to sound waves that are 17m and 17mm long in air at standard conditions. The relationship between wavelength and frequency is determined by the speed of the wave: $\lambda \cdot f = v$.
  - The speed of sound waves varies depending on the medium. In air at standard conditions, sound travels $v = 331 + 0.6 \cdot T$ m/s with T the temperature in Celsius. In water, sound travels much faster at about $v = 1482$ m/s. In solids, speeds are even higher, ranging from $v = 4000$ m/s in wood to $v = 12{,}000$ m/s in diamonds.
  - Sound pressure is the difference between the pressure in the medium and the pressure of the wave. It is usually measured in decibels using the formula: $L_p = 20 \cdot \log_{10}(p/p_{ref})$, where p is the sound pressure and p_{ref} is the reference pressure (20 μPa in air). The factor of 20 is used because we are comparing the squares of pressures, and the logarithm adds an extra factor of 2. The logarithmic scale is necessary because of the wide range of perception. 0 dB is the threshold of hearing, and sounds above 120 dB can cause permanent hearing loss.

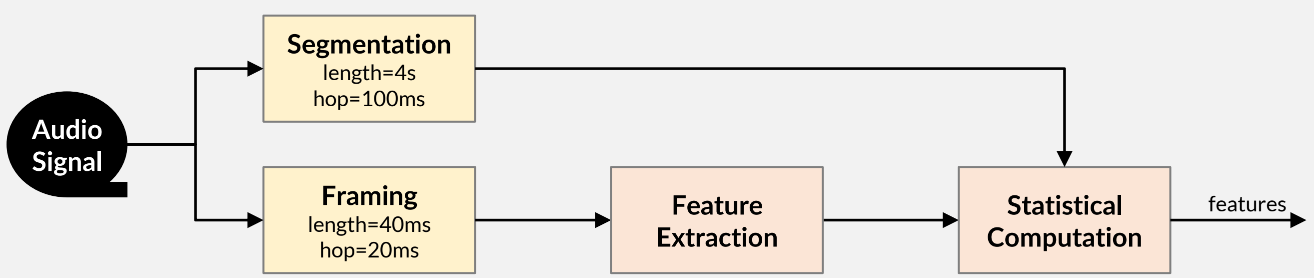
- **Human Perception** adds another dimension to the story. Humans hear only part of the full range of sound in nature. We respond to frequencies between 20 hertz and 20 kilohertz, while other species extend that range. Cats hear much higher pitches, and bats go well into ultrasound, using those sounds to navigate and hunt. Within the human range, several perceptual attributes shape how we experience sound.
  - **Pitch** is how we hear the frequency of sound. It helps us tell if music is high or low. Pitch needs a steady and clear frequency to stand out from noise. It's connected to frequency but it's not the same thing.
  - **Duration** is how long a sound is heard, from when it's first noticed until it fades away. It's connected to the actual length of the wave signal, but it can make up for breaks in the signal. For example, even if a radio signal is interrupted, it can still be heard as a continuous message.
  - **Loudness** is how loud or soft a sound seems. Our ears react to sounds over short periods of time, so a quick sound might seem quieter than a longer sound that's actually the same loudness. How loud a sound seems can change depending on the mix of different frequencies.
  - **Timbre** is the range of frequencies we hear over time. Different sound sources, like a guitar, a rock falling, or the wind, have distinct timbres that help us tell them apart. Timbre is like a fingerprint for sound, describing how it changes over time.
  - **Sonic Texture** is how different sounds interact, such as in an orchestra or on a train. The sound of a quiet market is very different from a busy party.
  - **Spatial Location** refers to where a sound is coming from, not necessarily its actual source. This, combined with the sound's quality, helps us to focus on one specific source, like a friend at a party.
- Sound is represented as an **amplitude signal** over time. To turn the continuous signal into a digital form, it is sampled at a fixed frequency f_s. The Nyquist-Shannon sampling theorem says that the sampling rate determines the highest frequency f_{max} that can be accurately represented, which is half the sampling rate ($f_{max} = f_s/2$). Since human hearing ranges from 20Hz to 20kHz, CDs use a sampling rate of 44.1kHz and DVDs use 48kHz.

- For many analysis tasks the time domain alone does not show how the spectrum changes. A waveform shows how pressure varies over time but not which frequency components produce that variation. When a sound contains several tones, harmonics, or noisy elements, those components blend into the amplitude curve and cannot be separated by sight. The frequency domain provides this missing view by breaking the signal into its component frequencies and showing how much energy each one carries.
- A standard Fourier transform breaks a signal down into its constituent frequencies and provides one global spectrum that averages frequency content from beginning to end. Any changes that occur during the recording are lost in that average. This matters when sound evolves over time, as most natural and musical sounds do. A music instrument may begin with a bright attack and then settle into a softer tone. A single Fourier transform cannot capture these transitions because it merges every moment into one fixed description.
- The **Short-Time Fourier Transform (STFT)** addresses this problem by applying a window function that isolates a small segment around a chosen time. Each windowed segment is transformed into a local frequency spectrum. By sliding the window along the signal, one obtains a record of how energy at different frequencies changes as the sound unfolds. The result is usually displayed as a spectrogram, where the squared magnitudes of the transform form a time frequency map. Window length controls the tradeoff between time and frequency resolution. A short window gives better timing but blurs frequency detail, while a long window gives better frequency detail but poorer timing.
  - The discrete frequency ω ranges from 0 to $f_{max} = f_s/2$ at steps of f_s/N Hz, with a window size of N samples. The absolute values of the complex value $X(t,\omega)$ indicate the magnitude of the frequency ω at time point t.
  - The image on the right shows the STFT with the red windowing function applied over time. The spectrogram then displays the squared magnitudes $|X(t,\omega)|^2$ over time. Different windowing functions can be used.

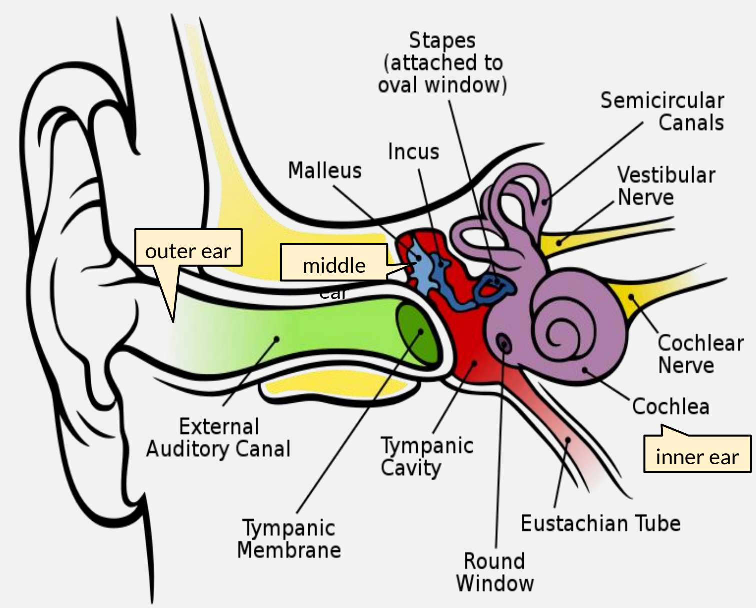
$$X(t,\omega) = \sum_{n=-\infty}^{\infty} x(n) \cdot w(n-t) \cdot e^{-i\omega n}$$



- To build features, the signal is split again, not to change it but to organize the information (compare with chunking in text retrieval). The audio is first divided into overlapping frames, and each frame yields a feature vector. Overlap prevents abrupt changes at frame boundaries and keeps feature changes smooth. Frames are then grouped into longer segments that can span several seconds. These segments serve as the basic units for retrieval or classification. Frame-level features are combined into segment-level descriptors by computing statistics that summarize each segment's behavior. A recording thus becomes a sequence of segments, each reflecting a local portion of the sound.
- Frame length and segment length depend on the task at hand.
  - **Frame size**: Assuming a sampling frequency of $f_s = 48$ kHz, a frame size of 40ms gives us $N = 1920$ samples. This means the frequency resolution of STFT is $\frac{f_s}{N} = 20.83$ Hz. This resolution is not enough to distinguish between two musical pitches in the middle octave, but it's okay for the first and second octave. To improve frequency resolution, we can increase the window size, but this will blur the spectrum and we lose precision along the time axis. With STFT, we have to compromise either on frequency resolution or time resolution.
  - **Segment size:** The size of the segment depends on the specific task. For detecting timbre (like guitar, rock falling, wind), a shorter segment is suitable. For spoken text, different segmentation methods can be used. The 4s shown in the picture is a good starting point for analyzing audio in general.
- With this basis we can extract perceptual features in both the time domain and the frequency domain. Time domain descriptors, like energy or envelope shape, estimate how loudness changes over time. Frequency domain descriptors, like spectral centroid, show the dominant frequencies and how they vary. To capture human perception, we must understand how the auditory system converts the physical amplitude signal and its spectrum into meaningful sensations.

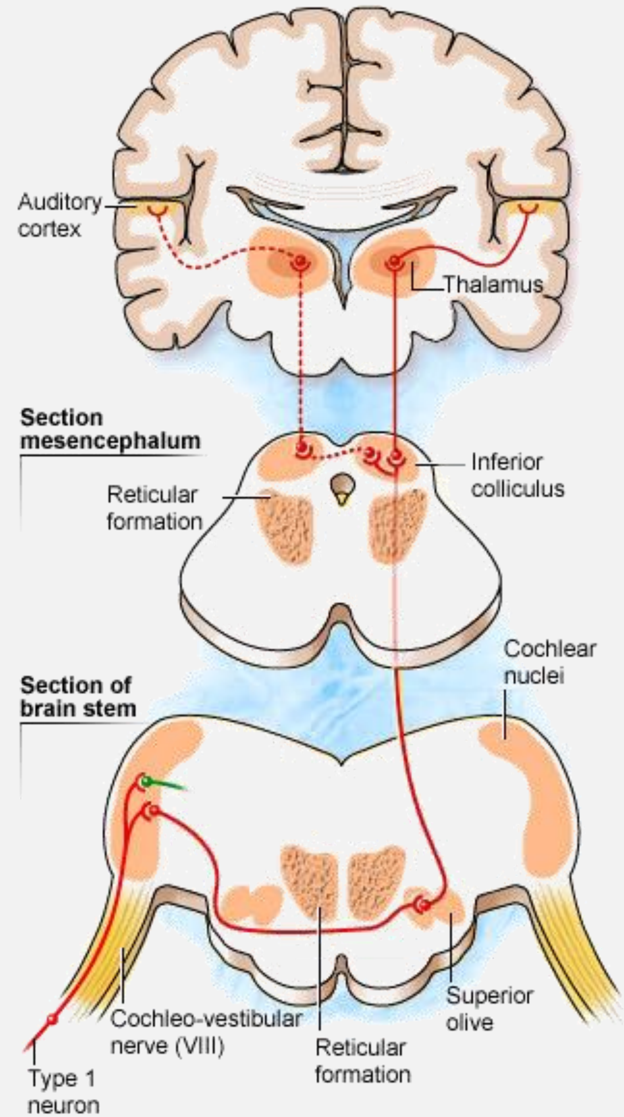


- The ear does not respond directly to raw pressure changes. It first filters sound with the outer ear, then amplifies and shapes it in the middle ear, and finally breaks the sound down by frequency along the basilar membrane in the inner ear. Different places on that membrane respond to different frequency ranges. The neural signals that result convey an organized version of the sound spectrum as we perceive it, not the raw spectrum.
  - The **outer ear** is the visible part of the ear. It reflects and amplifies sound and provides clues about where a sound comes from. Sounds enter the ear canal, which boosts frequencies near 3 kHz by as much as 100 times. This boost helps us recognize voices and hear differences like "s" versus "f". The amplified sound then reaches the eardrum.
  - Sound waves move from the eardrum into the air-filled **middle ear** and then through three tiny bones: the hammer (malleus), anvil (incus), and stirrup (stapes). These bones act like a lever to amplify the signal at the oval window, also called the vestibular window. This amplification is necessary because the cochlea is filled with fluid. A reflex in the middle ear helps protect against damage from very loud sounds.
  - The **inner ear** consists of the cochlea and the vestibular system.
    - The **vestibular system** helps with balance and sensing body movement.
    - Inside the **cochlea** is the organ of Corti that does most of the ear's work of turning mechanical vibrations into nerve signals. It has two kinds of hair cells, each with its own role.
    - The **outer hair cells** act like tiny biological amplifiers. They change shape when vibrations arrive and, in doing so, sharpen the motion of the basilar membrane. This extra boost helps the ear tell apart frequencies that are very close together and makes it more sensitive to quiet sounds.
    - The **inner hair cells** convert vibration of the membrane into electrical signals for the auditory nerve. Their response is not proportional to motion: they respond strongly to small movements but cut back their output when motion becomes too large. This protects the inner ear and shapes how we perceive loudness. High frequencies are detected near the base, close to the middle ear, while low frequencies are detected toward the apex, creating a natural frequency map.



Chittka L, Brockmann - Perception Space—The Final Frontier, A PLoS Biology Vol. 3, No. 4, e137 doi:10.1371/journal.pbio.0030137

- Outer hair cells in the cochlea actively amplify quiet sounds more than loud ones. This increases the ear's usable dynamic range. Chemical processes in the cochlea also let the ear adapt to steady or unchanging sounds. When the response to a constant tone fades slightly, new or changing sounds stand out more. Together, these mechanisms explain why the ear is both highly sensitive and highly selective, and why perceptual features must reflect these non-linear, adaptive properties.
- Inner hair cells convert mechanical vibrations into electrochemical signals by releasing neurotransmitters, which are then picked up by the auditory nerve fibers. The cochlear nerve contains about thirty thousand of these fibers, and each one responds best to a particular frequency at a particular loudness level. Alongside it runs the vestibular nerve, which carries information about balance and motion. Sound information reaches the brain through two routes: the primary auditory pathway and the reticular pathway. The reticular pathway blends all sensory input and helps the brain decide which events need attention. The primary pathway follows a more specialized sequence.
  - The **cochlear nuclear complex** is the first brain region to analyze the frequency, intensity, and duration of the incoming signal.
  - The **superior colliculus** in the midbrain uses differences across frequency bands to estimate the direction from which a sound arrives.
  - The thalamus, specifically the **medial geniculate body**, prepares the information for actions such as speaking or reacting to a sudden noise.
  - Finally, the **auditory cortex** handles both basic and advanced aspects of hearing. Its neurons are arranged by frequency in detailed maps that help identify sound sources and reduce distortions caused by reflections. It also processes the fine timing of sounds, which is essential for understanding speech and recognizing complex auditory patterns such as music.



12.2 Perceptual Features

- Perceptual features try to match how listeners actually hear sound. In the time domain, one looks at energy fluctuations and envelope shapes to capture sensations of loudness and dynamics. In the frequency domain, one tracks brightness with measures such as spectral centroid, the distribution of energy across bands, and the amount of spectral change over time. Psychoacoustic scales like Mel convert frequency into a form that better reflects human perception. After computing these descriptors frame by frame, they are summarized at the segment level using statistics such as means, variances, and temporal trends. This stabilizes the descriptors and makes them closer to the perceptual units humans tend to recognize.
- The first set of features describes audio files from an acoustic perspective across different domains.
  - **Time Domain** refers to analyzing the raw signal in terms of time and amplitude.
  - **Frequency Domain** transforms raw signal with STFT and analyzes frequencies and their energies at the given time point (see window technique)
  - **Perceptual Domain** is modelling the perceptual interpretation of the human ear
- After this chapter we will study musical features and fingerprinting techniques that often rely on these perceptual features:
  - Musical features describe the structure listeners hear in melody, rhythm, and harmony. Pitch tracking follows the fundamental frequency and its changes, whether from a single voice or mixed with other sounds. Tempo estimation finds the periodic pattern that sets the pace of a piece, while beat tracking marks the regular points that guide musical flow. Depending on the application, one may also extract higher level information such as key, chord sequences, or the typical timbres of instruments. These descriptors let systems treat sound not just as energy patterns but as organized artistic material.
  - Fingerprinting uses a different method. Instead of describing how sound is heard or arranged musically, it creates a small code that uniquely identifies a recording. A common approach is to find clear features in the sound spectrum and encode the relationships between them. These codes must resist background noise, compression artifacts, and playback distortions so that short or imperfect queries can still be matched reliably to a large database. Fingerprinting is the core technology behind many commercial music recognition systems.

12.2.1 Features in the Time Domain

- In the time domain we use the raw amplitude signal from a single frame (see segmentation). For example, with a sampling rate of 48 kHz and a frame length of 40 ms, the frame contains 1920 samples, and the hop between consecutive frames is 20 ms.
  - **Short-Time Energy (STE)** measures the total energy in a signal by adding up the squares of the values and then normalizing it by the length of the frame. In audio signals, power is typically measured in decibels, which is one-tenth of a bel, a unit used in the first telephone system. An increase of 10 dB represents a tenfold increase in power. The metric is logarithmic: $L_P = 10\log_{10}(P/P_0)$. With that, STE for an amplitude signal $x(t)$ within a frame F_i (with samples from $1 \leq t \leq N$) is defined as:

$$E_{STE}(i) = 10\log_{10}\left(\frac{1}{N}\sum_{t=1}^{N} x(t)^2\right)$$

  - **Zero-Crossing Rate (ZCR)** measures how frequently the amplitude signal changes from positive to negative values within a frame.

$$ZCR(i) = \frac{1}{2N}\sum_{t=2}^{N}|sgn(x(t)) - sgn(x(t-1))|$$

  - **Entropy of Energy (EoE)** measures sudden changes in the energy of the audio signal within a frame F_i. To do this, the frame is divided into smaller sub-frames of the same length that cover the entire frame. For each sub-frame, the energy is measured and then adjusted to the total energy of the frame to get a sequence of values that add up to 1. The entropy of these values is the Entropy of Energy. Choose L and N_{sub} such that $N = L \cdot N_{sub}$:

$$H_{EoE}(i) = -\sum_{l=1}^{L} e(i,l) \cdot \log_2 e(i,l) \qquad e(i,l) = \frac{\sum_{t=l\cdot N_{sub}}^{(l+1)\cdot N_{sub}-1} x(t)^2}{\sum_{t=1}^{N} x(t)^2}$$

- **Segment-level features in the time domain:** These features provide statistics for a segment S_j with M frames. For example, if the segment is 4 seconds long, with frames of 40 milliseconds and a frame hop distance of 20 milliseconds, there will be 199/200 frames, depending on how the last frame is treated as it is partially inside and partially outside the segment.
  - **Histograms** are created by dividing the range of a feature's values (STE, ZCR, EOE) into sections and counting how many values fall into each section across the frames of a segment. The resulting numbers are normalized to create a histogram. This method is not commonly used because it produces larger features than moments.
  - Calculate the **moments** for STE, ZCR and EOE values within the segment S_j. These features describe the distribution of values within the segment.
  - **Low Short-Time Energy Ratio (LSTER)** measures the percentage of frames in the segment S_j where the Short-Time Energy (STE) is less than one-third of the average STE across the segment. This is important because speech signals often have pauses between syllables, leading to greater variation.

$$r_{LSTER}(j) = \frac{1}{M}\sum_{i=1}^{M}\begin{cases}1 & E_{STE}(i) < \dfrac{\mu_{STE}(j)}{3} \\ 0 & \text{otherwise}\end{cases} \qquad \mu_{STE}(j) = \frac{1}{M}\sum_{i=1}^{M} E_{STE}(i)$$

  - The **High Zero-Crossing Rate Ratio (HZCRR)** measures the number of zero-crossings and counts how many frames have much higher zero-crossings than on average within the segment:

$$r_{HZCRR}(j) = \frac{1}{M}\sum_{i=1}^{M}\begin{cases}1 & ZCR(i) \geq \dfrac{3\mu_{ZCR}(j)}{2} \\ 0 & \text{otherwise}\end{cases} \qquad \mu_{ZCR}(j) = \frac{1}{M}\sum_{i=1}^{M} ZCR(i)$$

- The last two features are useful for distinguishing between speech and music signals. Speech has many frames with low energy, while music has consistently high energy throughout the segment. Additionally, speech has more frames with high zero-crossings because many frames have none due to energy pauses. On the other hand, music has a consistent rate of zero-crossings across frames, with only a few frames showing significantly more zero-crossings compared to the average frame in the segment.

12.2.2 Features in Frequency Domain

- We first analyze the Fourier transformed signal in the frequency domain using a single frame F_i (STFT). For example, with a sampling rate of $f_s = 48$ kHz and a frame size of 40ms, there are $N = 1920$ samples, and the hop distance between subsequent frames is 20ms. The Fourier transformed values $X(i, \omega)$ denotes the frequency spectrum of frame F_i with $0 \leq \omega \leq f_s/2$ and with steps $\Delta\omega = f_s/N = 25$ Hz. It's important to note that in the discrete notation of the Fourier transformed, only the first half of the values are needed as the second half is symmetrical ($X(i, k)$ with $0 \leq k < N/2$). In the following, we often use the discrete form $X(i, k) = X(i, \omega(k))$ with $\omega(k) = k \cdot f_s/N$.
  - **The Spectral Centroid (SC)** is the center of gravity of the spectrum. It's the weighted average frequency in the spectrum of the frame, with the energy (magnitude) as weights. The energy is the absolute values of the complex value of $X(i, k)$. For convenience, let $K = N/2 - 1$. Therefore:

    $$SC(i) = \frac{\sum_{k=0}^{K} \omega(k) \cdot |X(i, k)|}{\sum_{k=0}^{K} |X(i, k)|}$$

    The centroid describes the "sharpness" of the signal is in the frame, a weighted average of frequency and energy.
  - **Spectral Roll-off** (**SR**, ω_r) shows the frequency at which the total sum of energy with frequencies smaller than ω_r is $C = 85\%$ of the overall sum of energy. We are looking for a value $0 \leq r \leq K$.

    $$\omega_r = \omega(r) \quad \text{with } r \text{ smallest value that fulfills:} \quad \sum_{k=0}^{r} |X(i, k)| \leq \mathrm{C} \cdot \sum_{k=0}^{K} |X(i, k)|$$

    Related to the spectral centroid, it measures how skewed the spectrum is towards higher frequencies which are dominant in speech.
  - **Band-Level Energy (BLE)** is the total energy within a specific frequency range. This range is determined using a weighting function in the Fourier domain with $0 \leq k \leq K$. The feature value is measured in decibels to match how we perceive sound.

    $$BLE(i) = 10 \log_{10} \left(\sum_{k=0}^{K} |X(i, k)|^2 \cdot w(k) \right)$$

- **Spectral Flux (SF)** measures the squared differences in normalized magnitudes from the previous frame. It shows how the spectral content is changing locally. A high value means there is a sudden change in magnitudes, which can lead to a significant change in perception (only for $i > 1$).

$$SF(i) = \sum_{k=0}^{K} \left(\frac{|X(i,k)|}{\sum_{k=0}^{K} |X(i,k)|} - \frac{|X(i-1,k)|}{\sum_{k=0}^{K} |X(i-1,k)|} \right)^2$$

- **Spectral Bandwidth (SB)** measures the distance of frequencies from the spectral centroid. It shows how much the frequencies deviate from the spectral centroid

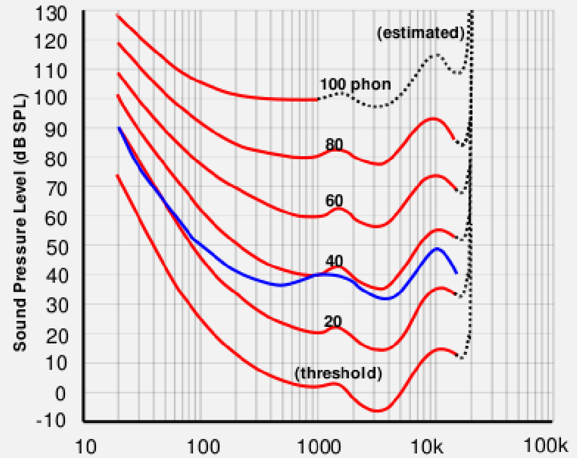
$$SB(i) = \sqrt{\frac{\sum_{k=0}^{K} |X(i,k)| \cdot \big(\omega(k) - SC(i)\big)^2}{\sum_{k=0}^{K} |X(i,k)|}}$$

- **Segment-level features in the frequency domain**: to summarize a segment, we can use moments and histograms over the frame values for the different features mentioned.
  - **Histograms** are created by dividing the range of a feature's values into sections and counting how many values fall into each section across the frames of a segment. The resulting numbers are then adjusted to create a histogram of the feature values. This method is not commonly used because it produces larger features than moments.
  - **Moments** can describe the distribution for the feature values within the segment S_j with mean value, standard deviation, skewness, and kurtosis. For band level energy, we can use different weight functions and also calculate the covariance between different bands.

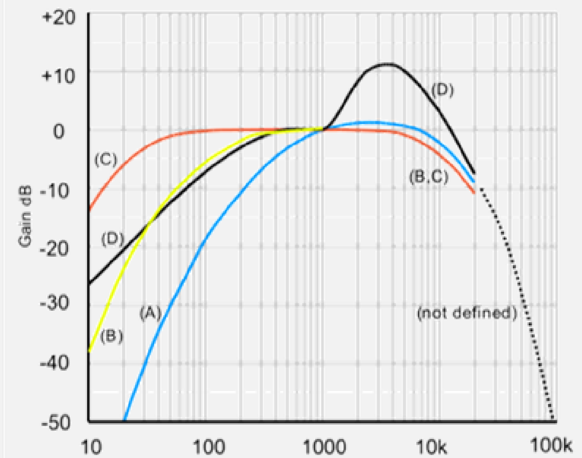
12.2.3 Psychoperceptual Features

- Human hearing and the way people perceive sound differ from simple physical measurements. For example, loudness reflects the energy in a sound wave, but people hear frequencies differently, especially between 2 and 5 kHz, which are important for understanding speech. The following features are designed to match the perception of a normal human ear:
  - **Loudness** is how the sound pressure of the wave is perceived by a “normal” person. Consider the top figure on the right side. The red curves show how much energy is needed for an average listener to hear a pure tone as equally loud as the frequencies increase from left to right. The needed energy drops a lot between 2 and 5 kHz because the outer ear is amplifying signals in this frequency range.
  - The international standard IEC 61672:2003 has different weighting functions to model this perception, as shown in the bottom right figure. The A-weighting curve (blue) is used the most, even though it's only good for quiet sounds.
  - The human perception of sound averages loudness over a 600-1000ms time period. So, the loudness for the frame F_i is the average over the previous 1000ms of the sound, not just the average of values in the current frame. Let O be the number of frames over the last 1000ms. For instance, with a hop size of 20ms, O=50. Loudness is measured in decibels to approximately match the logarithmic perception of a person:

$$L(i) = \frac{10}{O} \sum_{o=0}^{O-1} \log_{10} \left(\frac{1}{K} \sum_{k=1}^{K} A(k) \cdot |X(i-o,k)|^2 \right)$$



Equal-loudness contours (red) (from ISO 226:2003 revision)
Original ISO standard shown (blue) for 40-phons



A-weighting (blue), B (yellow), C (red), and D-weighting (blk)

– **Mel Frequency Cepstral Coefficients (MFCC)** show the energy spectrum over Mel frequency bands, which are similar to the human auditory system. The method involves 4 steps:

1. Fourier Transform: Calculate the Fourier transform over the frame F_i. Unlike the STFT, we do not use a windowing function. Let N be the number of samples in the frame F_i and f_s be the sampling rate (for example, $N = 1920, f_s = 48$ kHz).

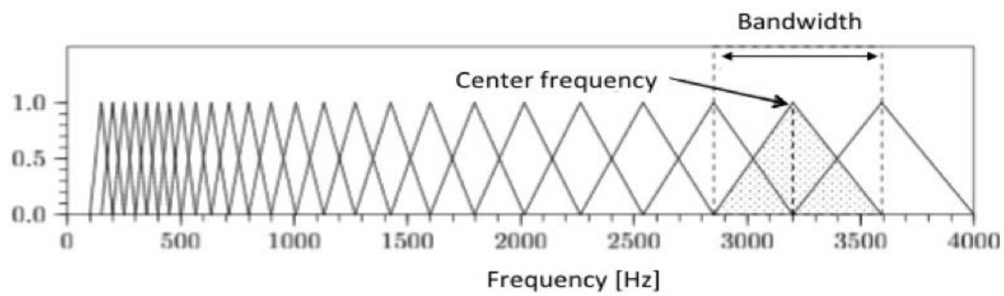
$$X(t,k) = \frac{1}{N}\sum_{j=0}^{N-1} x(j) \cdot e^{-i2\pi\frac{jk}{N}} \qquad \omega(k) = k \cdot \frac{f_s}{N}$$

2. Mel-Frequency Spectrum: We calculate the spectrum using Mel frequency bands. Usually, there are $B = 26$ bands, and we use $f_{lower} = 300$ Hz and $f_{upper} = 8000$ Hz to represent the lowest and highest frequencies. The conversion between frequencies and mels is as follows:

$$freq(m) = 700 \cdot \left(e^{\frac{m}{1125}} - 1\right) \qquad mel(f) = 1125 \cdot \ln\left(1 + \frac{f}{700}\right)$$

The bands are triangular window functions in the frequency space. Three frequencies mark the beginning, middle, and end points. Two bands overlap: the start point of one band is the middle point of the previous band. The frequencies are calculated in the Mel space to align with human perception. Given B bands, we need $B + 2$ frequencies given by $(0 \leq b \leq B + 1)$

$$f_c(b) = freq\left(mel(f_{lower}) + b \cdot \frac{mel(f_{upper}) - mel(f_{lower})}{B+1}\right)$$



With the frequencies $f_c(b)$, we can define the windowing function $d(b,k)$ over the Fourier coefficients $X(t,k)$ for a given time point t. The shape has a triangle form:

$$d(b,k) = \begin{cases} 0 & \text{if } \omega(k) < f_c(b-1) \\ \dfrac{\omega(k) - f_c(b-1)}{f_c(b) - f_c(b-1)} & \text{if } f_c(b-1) \leq \omega(k) \leq f_c(b) \\ \dfrac{\omega(k) - f_c(b+1)}{f_c(b) - f_c(b+1)} & \text{if } f_c(b) \leq \omega(k) \leq f_c(b+1) \\ 0 & \text{if } w(k) \geq f_c(b+1) \end{cases}$$

This finally allows us to calculate the Mel-frequency spectrum by adding up the magnitude values of the Fourier coefficients for each of the bands. This results in B values $M(t,b)$ for $1 \leq b \leq B$:

$$M(t,b) = \sum_{k=0}^{N/2-1} d(b,k) \cdot |X(t,k)|$$

3. Cepstral Coefficients: The cepstrum is like a spectrum of a spectrum. The updated version of MFCC calculates the coefficients using a discrete cosine transformation and uses the first half of the coefficients. If we started with $B = 26$, we obtain 13 cepstral values $c(t,b)$ with $1 \leq b \leq B/2$:

$$c(t,b) = \sum_{j=1}^{B} M(t,j) \cdot \cos\left(\frac{b(2j-1)\pi}{2B}\right) \qquad \text{with } 1 \leq b \leq B/2$$

4. Derivatives: The MFCC features are a mix of the cepstral $c(t,b)$ values and their first and second order derivatives. These derivatives show how the bands changes over time. By using 13 cepstral coefficients, we get 39 feature values.

$$\Delta c(t,b) = c(t+1,b) - c(t-1,b)$$
$$\Delta\Delta c(t,b) = \Delta c(t+1,b) - \Delta c(t-1,b)$$

$$MFCC(t) = [c(t,1), \dots, c(t,B/2), \Delta c(t,1), \dots, \Delta c(t,B/2), \Delta\Delta c(t,1), \dots, \Delta\Delta c(t,B/2)]$$

- **Segment-level features in the psychoperceptual domain**: We can calculate the moments or histograms of the perceptual features in a segment, just like we did before. For example, the standard deviation of the second MFCC coefficient $c(t,2)$ is very useful for telling the difference between speech and music.
  – MFCCs are commonly used in speech recognition. These feature values are employed in Hidden Markov Models or neural networks to recognize phonemes. Typically, cepstral coefficients from a large spoken text are clustered using a k-means clustering approach. These clusters are then used to quantize the vector and create states. The machine learning method then maps a series of state transitions to a phoneme. The phoneme stream is further processed to create words.

12.3 Musical Features

- Chroma based features are closely connected to the 12 different pitch classes in music {C, C♯, D, D♯, E ,F, F♯, G, G♯, A, A♯, B}. Each pitch class, like C, represents all the possible pitches at all octaves. All pitches are related to each other by octave. If two pitches of the same class are an octave apart, their frequency has a ratio of 1:2 (or 2:1), meaning that with each higher octave, the frequency doubles. Another important concept in music theory is the partials, overtone, fundamental, and harmonics.
  - Every musical instrument creates a mix of sine waves, known as **partials**. These partials have their own frequencies and vary in amplitude over time and thus giving each instrument its unique sound pattern, or timbre. Our ears are highly skilled at identifying and differentiating between different timbres, allowing us to distinguish between various instruments and voices in audio recordings.
  - The **fundamental** is the partial with the lowest frequency and corresponds to the pitch we hear. Harmonics are frequencies that are multiples of the fundamental frequency. Instruments may have both harmonic and inharmonic partials, but they are usually designed so that all partials are close to harmonic frequencies.
  - **Overtone** refers to all the partials except for the fundamental. The relative strength of the overtones determines the unique sound of an instrument as the sound changes over time.

  The standard pitch **A440** (also known as A4 of Stuttgart pitch) sets the A note of middle C at $f_{A4} = 440$ Hz and is used as a tuning standard for musical instruments. If we label the pitch classes (also called semitones) with $n = 0$ (C), …, $n = 11$ (B), we can express the frequency of the semitones in the octave o with $-1 \leq o \leq 9$ (MIDI numbers are $12(o + 1) + n$; A4 has number 69).

$$f_{A440}(o, n) = f_{A4} \cdot 2^{\frac{12o+n-57}{12}} = 440 \cdot 2^{\frac{12o+n-57}{12}}$$

Table of note frequencies (standard piano key frequencies)

Octave → Note↓	$o=-1$	$o=0$	$o=1$	$o=2$	$o=3$	$o=4$	$o=5$	$o=6$	$o=7$	$o=8$	$o=9$
C ($n=0$)	8.176	16.352	32.703	65.406	130.81	261.63	523.25	1046.5	2093.0	4186.0	8372.0
C♯ / D♭ ($n=1$)	8.662	17.324	34.648	69.296	138.59	277.18	554.37	1108.7	2217.5	4434.9	8869.8
D ($n=2$)	9.177	18.354	36.708	73.416	146.83	293.66	587.33	1174.7	2349.3	4698.6	9397.3
E♭ / D♯ ($n=3$)	9.723	19.445	38.891	77.782	155.56	311.13	622.25	1244.5	2489.0	4978.0	9956.1
E ($n=4$)	10.301	20.602	41.203	82.407	164.81	329.63	659.26	1318.5	2637.0	5274.0	10548.1
F ($n=5$)	10.914	21.827	43.654	87.307	174.61	349.23	698.46	1396.9	2793.8	5587.7	11175.3
F♯ / G♭ ($n=6$)	11.563	23.125	46.249	92.499	185.00	369.99	739.99	1480.0	2960.0	5919.9	11839.8
G ($n=7$)	12.250	24.500	48.999	97.999	196.00	392.00	783.99	1568.0	3136.0	6271.9	12543.9
A♭ / G♯ ($n=8$)	12.979	25.957	51.913	103.83	207.65	415.30	830.61	1661.2	3322.4	6644.9	
A ($n=9$)	13.750	27.500	55.000	110.00	220.00	440.00	880.00	1760.0	3520.0	7040.0	
B♭ / A♯ ($n=10$)	14.568	29.135	58.270	116.54	233.08	466.16	932.33	1864.7	3729.3	7458.6	
B ($n=11$)	15.434	30.868	61.735	123.47	246.94	493.88	987.77	1975.5	3951.1	7902.1	

Source: https://en.wikipedia.org/wiki/Scientific_pitch_notation

- To find the pitch in audio files, we need to extract the fundamentals. One way to do this is to match all frequencies from the STFT to a chroma value that represents the pitch class. We use the A440 standard with a frequency of $f_{ref} = 440$. Let's $\omega(k) = k \cdot f_s / N$ be the frequency mapping of the k-th Fourier coefficient with a sampling rate of f_s and with N samples. Then, we can find the chroma value, that is pitch class $p(k)$ and octave $o(k)$, as follows:

$$p(k) = \left\lfloor 9.5 + 12 \log_2 \left(\frac{k \cdot f_s}{N \cdot f_{ref}} \right) \right\rfloor \bmod 12 \qquad o(k) = \left\lfloor \frac{1}{12} \left(9.5 + 12 \log_2 \left(\frac{k \cdot f_s}{N \cdot f_{ref}} \right) \right) \right\rfloor$$

  - To create a chroma histogram, we add up the power spectrum using the mentioned mappings to determine the pitch class and octave. Then, we generate a histogram vector for each frame.

$$h_{chroma}(i, o, p) = \frac{1}{\sum_{k=0}^{K} |X(i,k)|^2} \cdot \sum_{k=0}^{K} \begin{cases} |X(i,k)|^2 & \text{if } o = o(k) \wedge p = p(k) \\ 0 & \text{otherwise} \end{cases}$$

  - Yet, this method does not allow us to capture the main pitch contour (or multiple pitches if the music is polyphonic). Instead, it only gives us a mapping to chroma values. We can estimate the fundamental frequency f_0 in a specific time window by finding the frequency that maximizes the sum of magnitudes across all its harmonics.

$$f_0 = \frac{f_s}{N} \cdot \underset{0 \le k \le K}{\operatorname{argmax}} \left(\sum_{m=1}^{M} g(k,m) \cdot |X(i, km)| \right) \qquad g(k,m) = \frac{\omega(k) + 27}{\omega(km) + 320} = \frac{k \frac{f_s}{N} + 27}{km \frac{f_s}{N} + 320}$$

    $g(k,m)$ is based on evidence and is used to determine the importance of each harmonic. The number M depends on the highest frequency in the spectrum and represents the number of harmonics being considered.

– With the fundamental f_0, we obtain the pitch class $p(f_0)$ and the octave $o(f_0)$ of the time window. To get multiple fundamentals from the frame, we repeat these steps:
  1. Compute the magnitude spectrum $\left|X^{(0)}(i,k)\right|$
  2. Iterate $t = 0, 1, \ldots$ as long as $\sum_{k=0}^{K}\left|X^{(t)}(i,k)\right| > \epsilon$
     – Compute f_0 on the magnitude spectrum $\left|X^{(t)}(i,k)\right|$
     – Modify the magnitude spectrum by subtracting the magnitudes of the harmonics from the computed fundamental f_0 to get the values for the next iteration $\left|X^{(t+1)}(i,k)\right|$

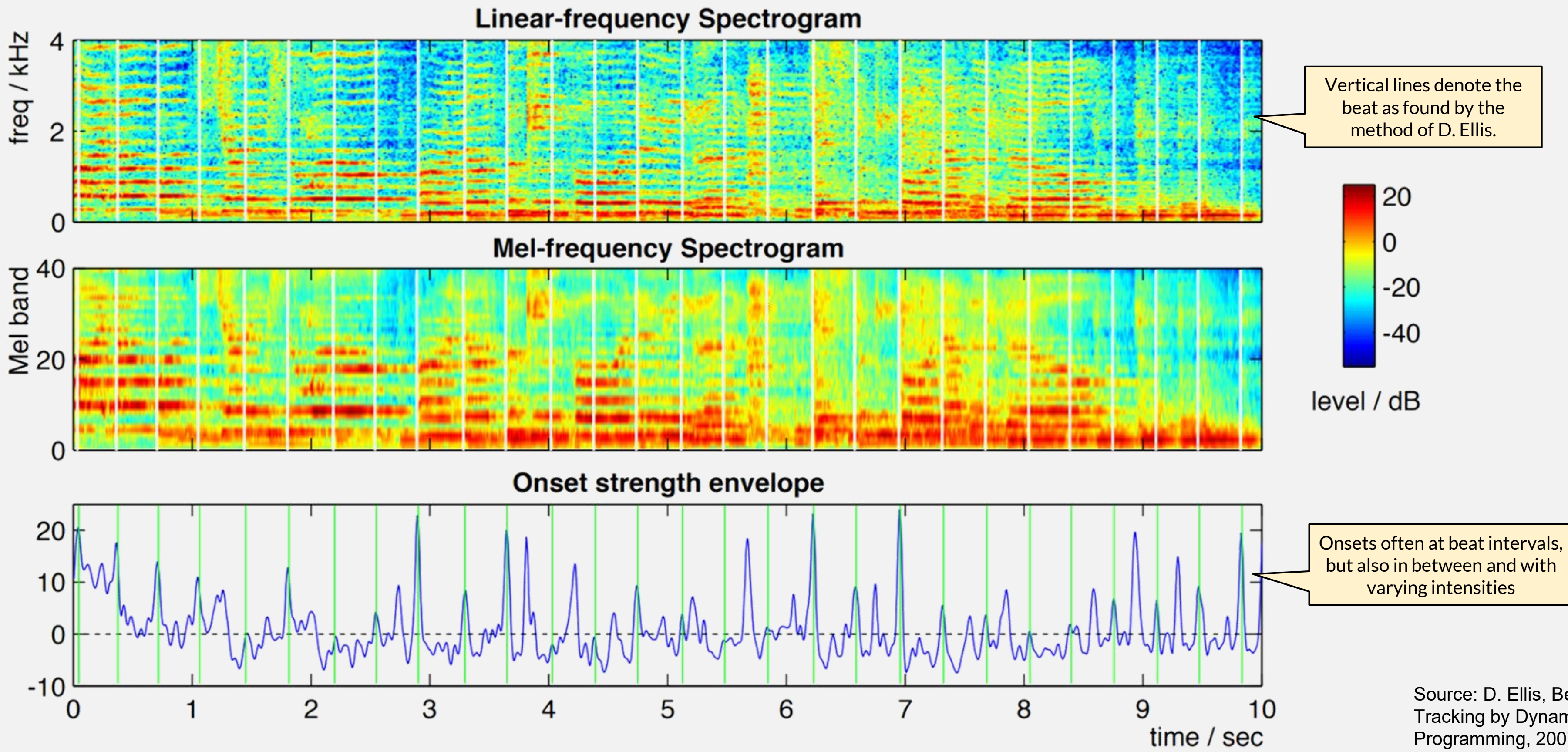
- We can alternatively calculate the fundamental frequency f_0 in the time domain. To do this, we calculate the autocorrelation of the audio signal at different time shifts Δt. Let N be the size of a frame and f_s be the sampling rate. To narrow down the search, we set the condition $1/f_{min} \geq \Delta\mathrm{t} \geq 1/f_{max}$ for the minimum and maximum frequency range for the fundamental f_0: $f_{min} \leq f_0 \leq f_{max}$. Additionally, time shifts are whole number multiples of the sampling period: $\Delta t = m/f_s$ with $f_s/f_{min} \geq m \geq f_s/f_{max}$. The autocorrelation for the frame F_i and the lag m is then defined as follows:

$$R(i,m) = \sum_{t=m}^{N} x(i,t) \cdot x(i,t-m)$$

  To find the basic element, we look for the lag m_0 that gives us the highest autocorrelation and then calculate the frequency from this lag.

$$f_0 = \frac{f_s}{m_0} \qquad m_0 = \underset{f_s/f_{min} \geq m \geq f_s/f_{max}}{\operatorname{argmax}} R(i,m)$$

- Another important aspect of music is the tempo, which is measured in beats per minute (bpm). In classical music, different tempos are defined by ranges such as Largo (40-60 bpm), Larghetto (60-66 bpm), Adagio (66-76 bpm), Andante (76-108 bpm), Moderato (108-120 bpm), Allegro (120-168 bpm), Presto (168-200 bpm), and Prestissimo (200+ bpm). These tempos can change throughout a piece. In pop music, the tempo is usually more consistent, with bpm ranging from 60 to 160, and 120 bpm being the most common choice.
  - Beat tracking is the process of finding the regular bursts of energy at the beat intervals. For a song with a tempo of 100 beats per minute, we would expect to see a spike in energy every 10 milliseconds (give or take, depending on how precise the musician is) to indicate the beats. However, it's not always as simple as that, as the example below demonstrates.



Source: D. Ellis, Beat Tracking by Dynamic Programming, 2007.

- Calculation of onset envelope: The onset is the positive slope on the energy over the spectrum at a specific time. Using STFT and mel bands, we get the mel spectrum $|X_{mel}(i,b)|$ as a weighted function from the frequency spectrum. The weighting ensures that the areas under the triangular mel bands are equal. This enhances the resolution of lower frequencies and emphasizes them over higher frequencies (essentially weighting by the inverse of the band width). The onset is similar to the spectral flux, but only positive slopes (onsets) are considered and then converted to decibels. If B is the number of mel bands and is the current frame F_i, then the onset $o(i)$ is as follows:

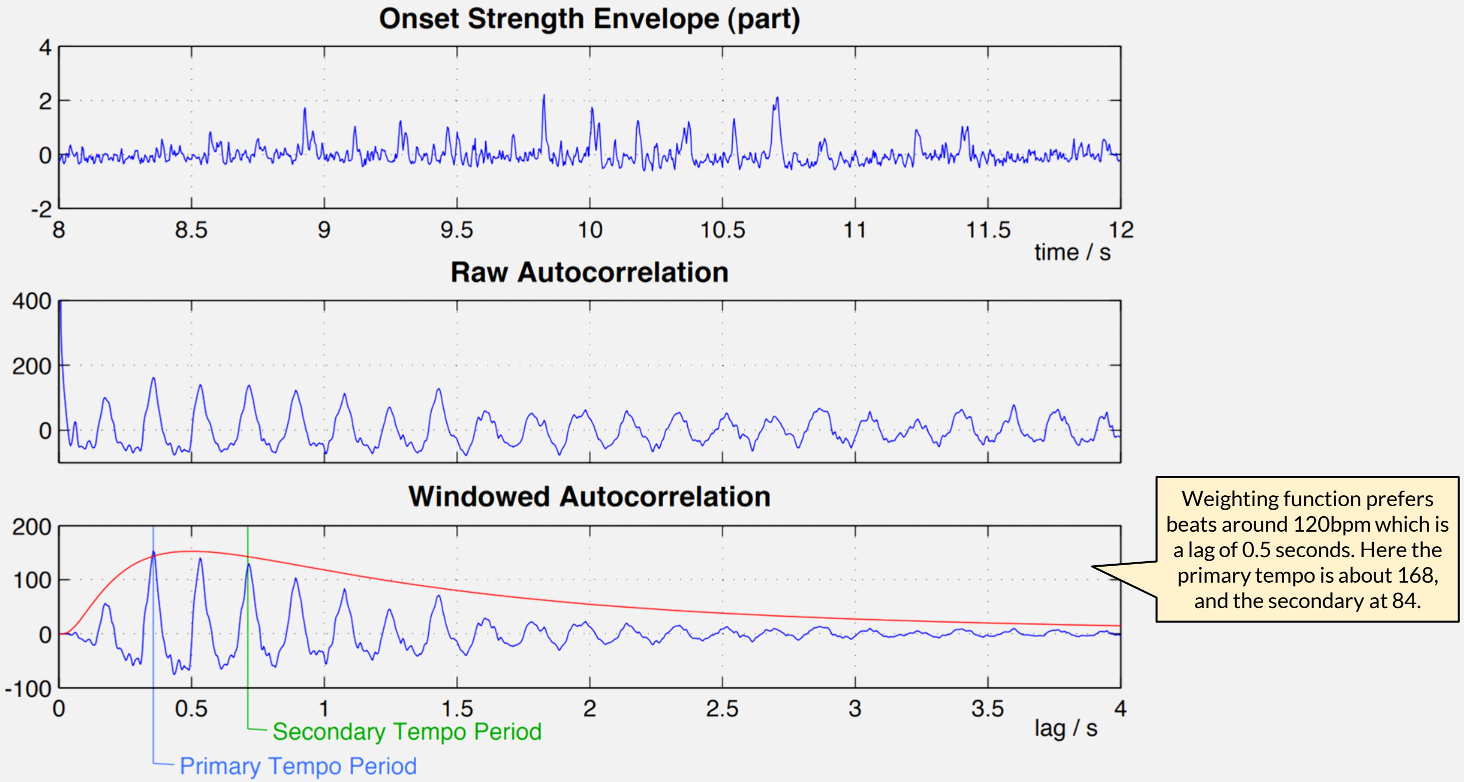
$$o(i) = \sum_{b=1}^{B} \max\left(0, \frac{\log_{10}|X_{mel}(i,b)|}{\log_{10}|X_{mel}(i-1,b)|} - 1\right)$$

- We estimate the global tempo by analyzing the onset $o(i)$ using a window function $w(i)$, like the Hann function. Basically, we're trying to find a time delay Δt where the peaks in the onset function line up. The time delay that gives us the highest autocorrelation is the global tempo. We can calculate the tempo for each frame to create a tempogram, which shows autocorrelations for each frame F_i, the lag l, and the window function $w(i)$.

$$a(i,l) = \sum_{j=0}^{W} w(j) \cdot o(i+j) \cdot o(i+j+l)$$

- The tempo is determined by the lag l_0 with the strongest correlation, and we can convert it to beats per minute with $\Delta t = l_0/f_s$ and so the tempo is $\frac{60}{\Delta t} = 60\frac{f_s}{l_0}$ bpm. Sometimes, we also see other peaks at $\{0.33l_0,\ 0.5l_0,\ 2l_0,\ 3l_0\}$ which indicate secondary tempos if their correlation is high enough. Additionally, we may prefer beats around 120 bpm if we know the genre (for example, pop).
- See visualization on the next page.

Example for tempo estimation within a time frame:



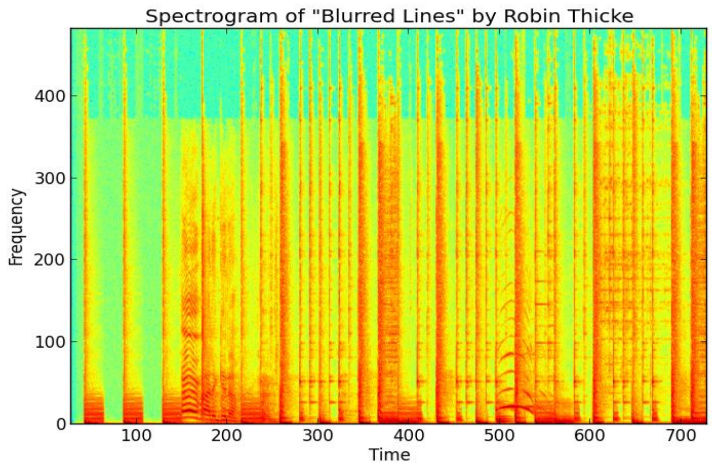
– **Beat tracking** is the process of identifying the time points $\{t_i\}$ when the onsets occur in the audio signal, similar to how a person would tap along to the beat. The goal is to find these time points so that they match the tempo of the music, with some small variations allowed. This is done by optimizing an objective function $F(t_i - t_{i-1}, l_0)$, which includes a penalty for deviations from the ideal tempo and a weighting to balance the importance of the onset values and penalty values.

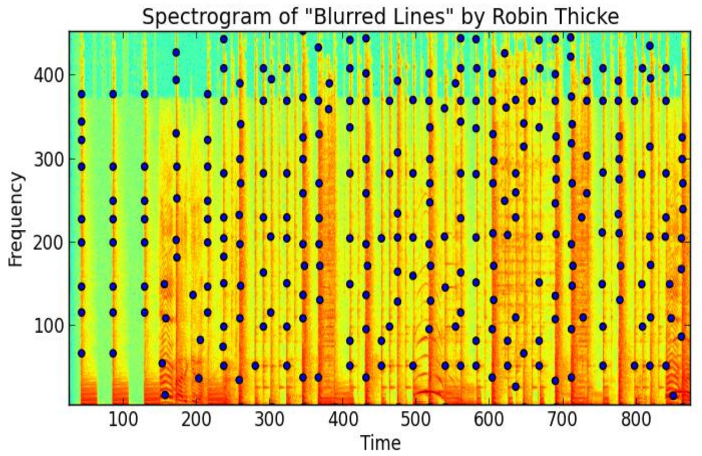
$$C(\{t_i\}) = \max_{\{t_i\}} \left(\sum_{i=1}^{T} o(f_s \cdot t_i) + \alpha \sum_{i=2}^{T} F(f_s \cdot (t_i - t_{i-1}), l_0) \right) \qquad F(\Delta l, l_0) = -\left(\log \frac{\Delta l}{l_0} \right)^2$$

12.4 Fingerprinting for Music Recognition

- Identifying a song from a real-world audio recording presents fundamental challenges that the previous approaches cannot overcome easily. When you record a song in a noisy place, such as a restaurant with conversation, a crowded bar, or a moving vehicle, the captured audio is very different from the studio-quality reference recording in a database. Background noise, compression artifacts, and acoustic distortion all change the audio signal, so you cannot simply compare waveforms or spectral envelopes directly.
- The main challenge is achieving robustness by finding **invariant features**. A song's recognizable core, its melody and harmony, survives many distortions, but that core must be extracted so it stays stable across different kinds of degradation. Traditional frequency-domain analysis keeps too much detail: amplitude, phase, and exact frequency positions all change with noise and recording conditions. A fingerprinting system must use features that resist these changes while remaining specific to each song.
- Audio fingerprinting is a **content-based identification method** that lets apps like Shazam recognize songs from short real time recordings, even with a lot of noise or distortion. Instead of comparing whole audio files, it extracts small features that resist noise and capture a song's frequency and timing patterns. This allows identification from a brief microphone recording in a busy restaurant or while driving. Modern audio fingerprinting works because reference recordings are pre-cataloged. Systems preprocess each song's official recordings to create a compact fingerprint. When a query recording arrives, the app makes a fingerprint from it and searches the database for the best matches.
- This approach immediately suggests its fundamental limitations:
  - **Singing Your Own Version**: Fingerprinting systems are built to match pre-recorded, cataloged recordings. When you sing a song yourself, you introduce large changes in pitch stability, vibrato, timing, and spectral content. Your voice produces a very different frequency pattern than the original track. Fingerprint extraction relies on features like instrumental arrangement and precise timing, which change a lot in a vocal performance.
  - **Live Recordings:** Live performances differ from studio recordings in ways that exceed fingerprinting's robustness. They can use different instruments, change tempo, include extended improvisations, and contain substantial audience noise. Shazam can sometimes recognize live versions when they stay close to the original, but most live performances differ too much in their time and frequency characteristics to reliably match the fingerprints in the database. The system was trained on the assumption that recordings preserve the song's core frequency structure, an assumption live performances often violate.

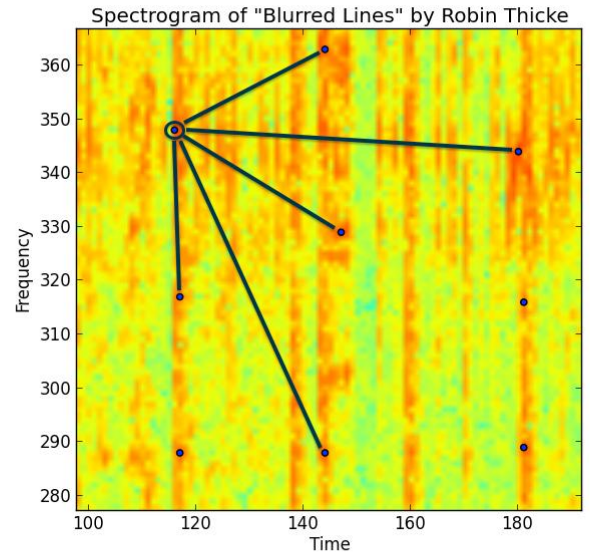
- **Step 1: Computing the Spectrogram**
  - We first transform the audio into a spectrogram using the Short-Time Fourier Transform (STFT). The recording is split into overlapping short time windows. For each window, the FFT shows which frequencies are present and their amplitudes. The result is a 2D image where the horizontal axis is time, the vertical axis is frequency, and color shows the energy at each time and frequency point.
  - The spectrogram is the key intermediate representation because it reveals the noise problem. Pure background noise shows up as uniform energy across many frequencies and time frames, while the signal of interest, the actual song, appears as localized peaks of high energy concentrated at specific frequencies and times.
- **Step 2: Identifying Constellation Map Peaks**
  - Next is peak detection. This step finds peaks in the spectrogram that correspond to the most prominent frequency components. Instead of using every frequency value, which would create a huge, noise-sensitive fingerprint, Shazam keeps only the local maxima: points whose energy is higher than all neighbors in both time and frequency.
  - This produces a sparse set of points, called the constellation map, that characterizes the song's structure. Background noise produces random peaks across the spectrogram, but these noise peaks are usually weaker and less concentrated. The strongest peaks, those that meet the local-maximum criterion, tend to match real signal features such as instrument harmonics. Isolated weaker noise peaks are filtered out by the peak detection process, even when background noise is high.
  - The result is a noise-robust representation that captures the song's main frequency structure while removing high-frequency noise and weaker spectral features.





source: https://willdrevo.com/fingerprinting-and-audio-recognition-with-python/

- **Step 3: Combinatorial Hashing**
  - Constellation maps are much smaller than full spectrograms, but they still contain many isolated peaks, so direct matching is slow. A 5 to 10 second query can produce hundreds or thousands of peaks. Comparing each of those to billions of stored peaks across millions of songs would be computationally expensive.
  - Shazam uses combinatorial hashing. Instead of storing or matching single spectral peaks, it makes fingerprint hashes by pairing peaks. The procedure is:
    1. **Anchor point selection**: Each peak in the constellation is chosen as an anchor point.
    2. **Target zone definition**: For each anchor point, a target zone is set in the time-frequency space after the anchor point, for example within the next 20 to 40 frequency bins and the next 50 to 200 ms.



source: https://willdrevo.com/fingerprinting-and-audio-recognition-with-python/

    3. **Hash generation**: The anchor point is paired with every other peak inside its target zone. For each pair the algorithm records the first peak's frequency, the second peak's frequency, and the time difference between them.
    4. **Hash encoding**: These three values are combined into a single compact hash, typically a 32-bit integer, using quantification. The hash serves as a fingerprint for that pair relationship.
  - This method is clever because it turns the matching task into simple hash table lookups much like inverted files for classical text retrieval. Instead of comparing thousands of individual peaks, the system only looks for hashes that appear in both the query fingerprint and the database fingerprints. Because each hash encodes relative frequency and timing, the same pair relationships will produce identical hashes even if peak positions move slightly from noise or distortion.
  - The combinatorial structure gives inherent robustness. If a few peaks are lost to noise and removed from the constellation, many hash relationships stay the same because they involve other peaks. If a few noise peaks are added, they mainly produce hashes that are not in the database, so they do not match and are ignored.

- **Step 4: Query Fingerprinting**
  - When a user taps the Shazam button, the system records a 5 to 10 second audio sample and processes it the same way as catalog songs. The result is a query fingerprint: a set of hashes that represent the recording's acoustic structure.
- **Step 5: Database Matching and Score Aggregation**
  - **Hash Matching:** Shazam looks up each hash from the query fingerprint in a table that holds every hash from the song catalog. The table also stores metadata showing which song each hash belongs to and the hash's position (time) in that recording. For each query hash, the system retrieves all matching entries from the database.
  - **Score Aggregation:** The main point is that many query hashes will match many catalog songs by chance or because of similar harmonic patterns. What matters is that hashes from the correct song will line up in time. For each candidate song:
    - 1. For each matching pair of hashes (query and database hash), compute the **time offset** as $t_{track} - t_{sample}$. This offset shows where in the original song the query sample most likely occurs.
    - 2. Build a **histogram** of the candidate song's time offsets. If the query comes from that song at a specific position, most matching hashes will vote for the same offset.
    - 3. The **score for the song** is the height of the largest histogram bin, the number of hashes that agree on a single time offset.
    - 4. The song with the **highest score** is returned as the match.
- This voting mechanism effectively filters out noise and false matches. Accidental hash collisions happen at random time offsets and make a flat histogram. True matches group their votes at a specific offset and make a clear peak.

False Positive

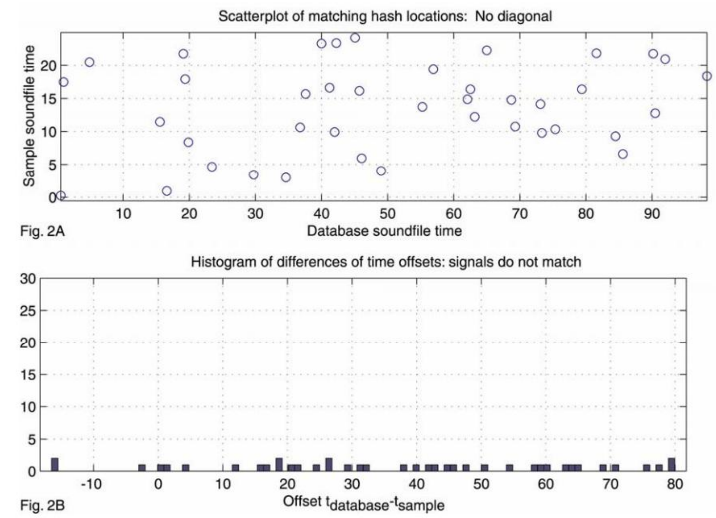


Fig. 2A

Fig. 2B

True Positive

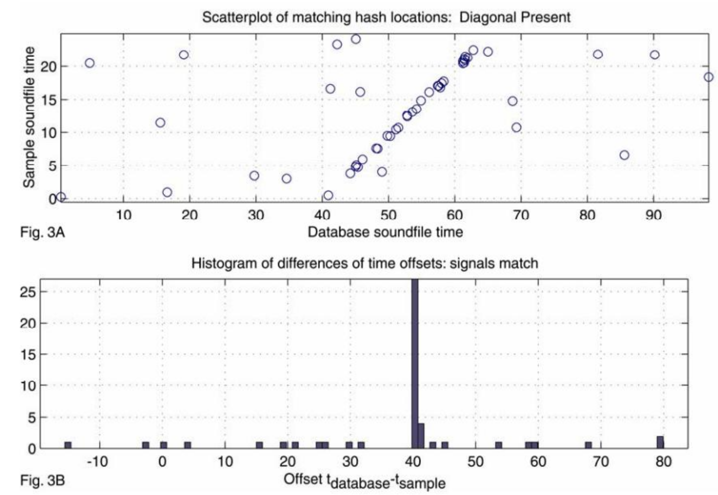


Fig. 3A

Fig. 3B

source: https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

12.5 Literature and Links

Feature Extraction

- T. Bäckström et al., **Introduction to Speech Processing**, https://speechprocessingbook.aalto.fi/index.html
- C. MacLeod, **abracadabra: How does Shazam work?**, 2022, https://www.cameronmacleod.com/blog/how-does-shazam-work
- W. Drevo, **Audio Fingerprinting**, 2013, https://willdrevo.com/fingerprinting-and-audio-recognition-with-python/
- A. Wang, **An Industrial-Strength Audio Search Algorithm**, 2003, https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf

Spectrograms Online

- https://borismus.github.io/spectrogram/
- https://musiclab.chromeexperiments.com/spectrogram/
- https://spectrogram.sciencemusic.org

Frameworks and Libraries

- **OpenSoundscape:** https://opensoundscape.org/ is a Python library for audio manipulation and analytics
- **Librosa** (http://librosa.github.io/librosa/) is a Python library for advances audio and music analysis. It provides base algorithms to create music retrieval systems.
- **OpenAI Whisper** (https://github.com/openai/whisper) is a transformer based transcriber

Interesting courses at other universities

- **Music Information Retrieval,** Vienna University of Technology, Austria, http://www.ifs.tuwien.ac.at/mir/
- **Music Information Retrieval,** New York University, USA, http://www.nyu.edu/classes/bello/MIR.html
- **Music Signal Processing,** Columbia University, USA https://www.ee.columbia.edu/~dpwe/e4896/index.html