

音楽データの基礎

Deep-people #2

- 深層学習のその前に、解析対象をよく知ろう！
 - $y = f(X)$ の X （と y も）をよく分かってから
- 音響データ
 - 音の性質と、フーリエ変換、スペクトログラムまで
- 楽譜データ
 - 音楽理論については割愛
 - コンピュータで扱う方法論が中心

- 多くをこちらからとっています

https://www.audiolabs-erlangen.de/content/05-fau/professor/00-mueller/04-bookFMP/02-slides/Mueller_FMP_Chapter1.pdf

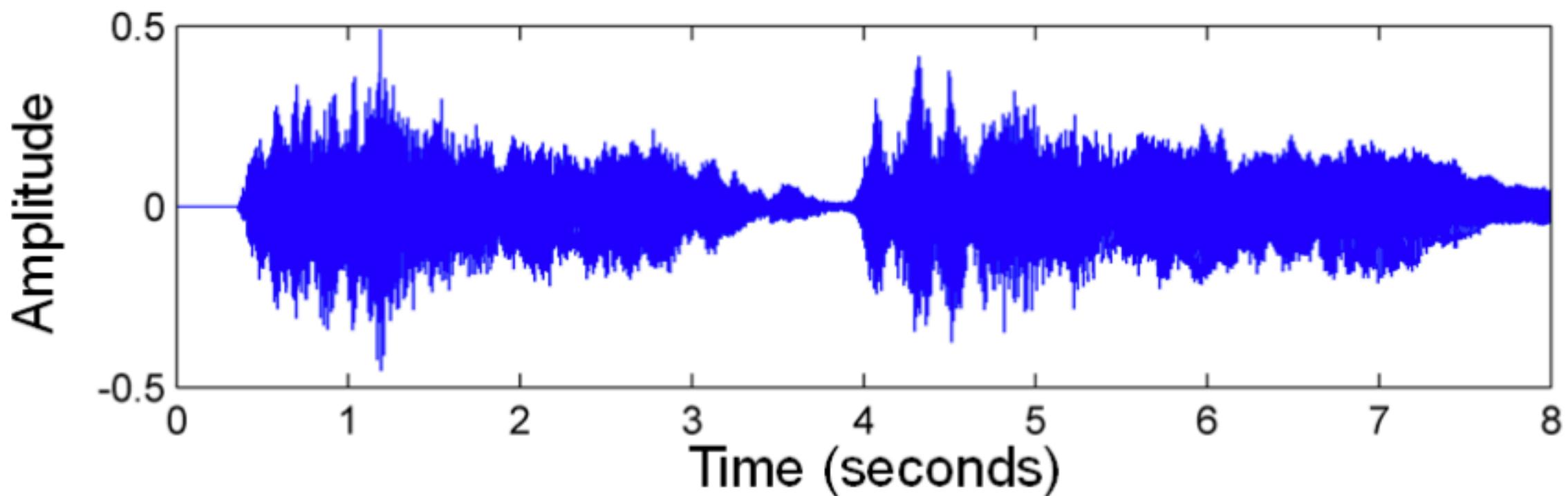
<https://www.microsoft.com/en-us/research/uploads/prod/2021/10/Tutorial-on-AI-Music-Composition-@ACM-MM-2021.pdf>

[https://mac.kaist.ac.kr/~juhan/gct634/Slides\[week1-3\]%20audio%20data%20representations.pdf](https://mac.kaist.ac.kr/~juhan/gct634/Slides[week1-3]%20audio%20data%20representations.pdf)

- 特に引用なく載せた図はこちらのいずれかからピックアップしています

データの形式

音響



楽譜

Allegro con brio ($\text{♩} = 108$)



音響

楽譜

データの性質

手に入る量

情報量

処理方法が近い・手法が参考になるモダリティ・分野

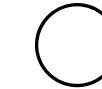
信号データ

大量

画像処理・センサデータ

符号データ

少量

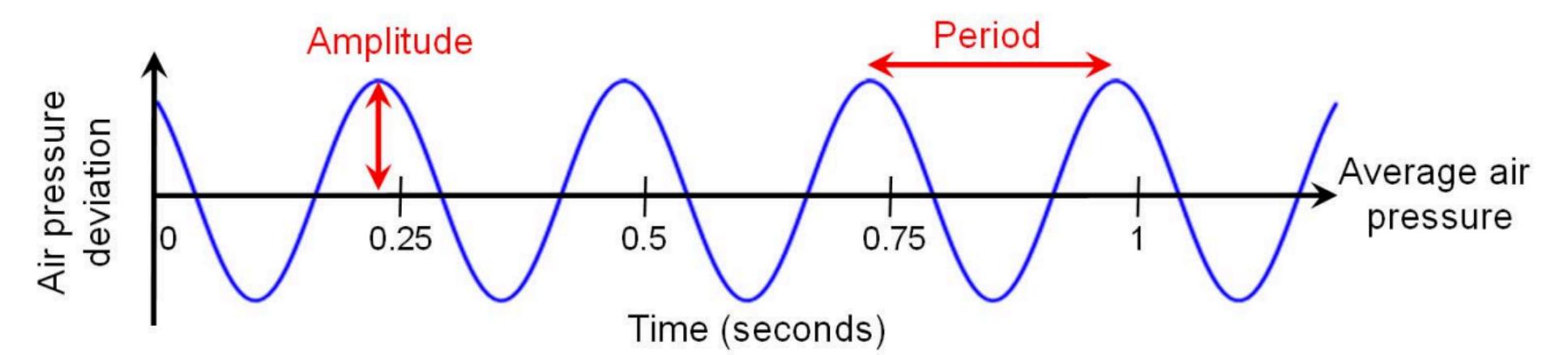
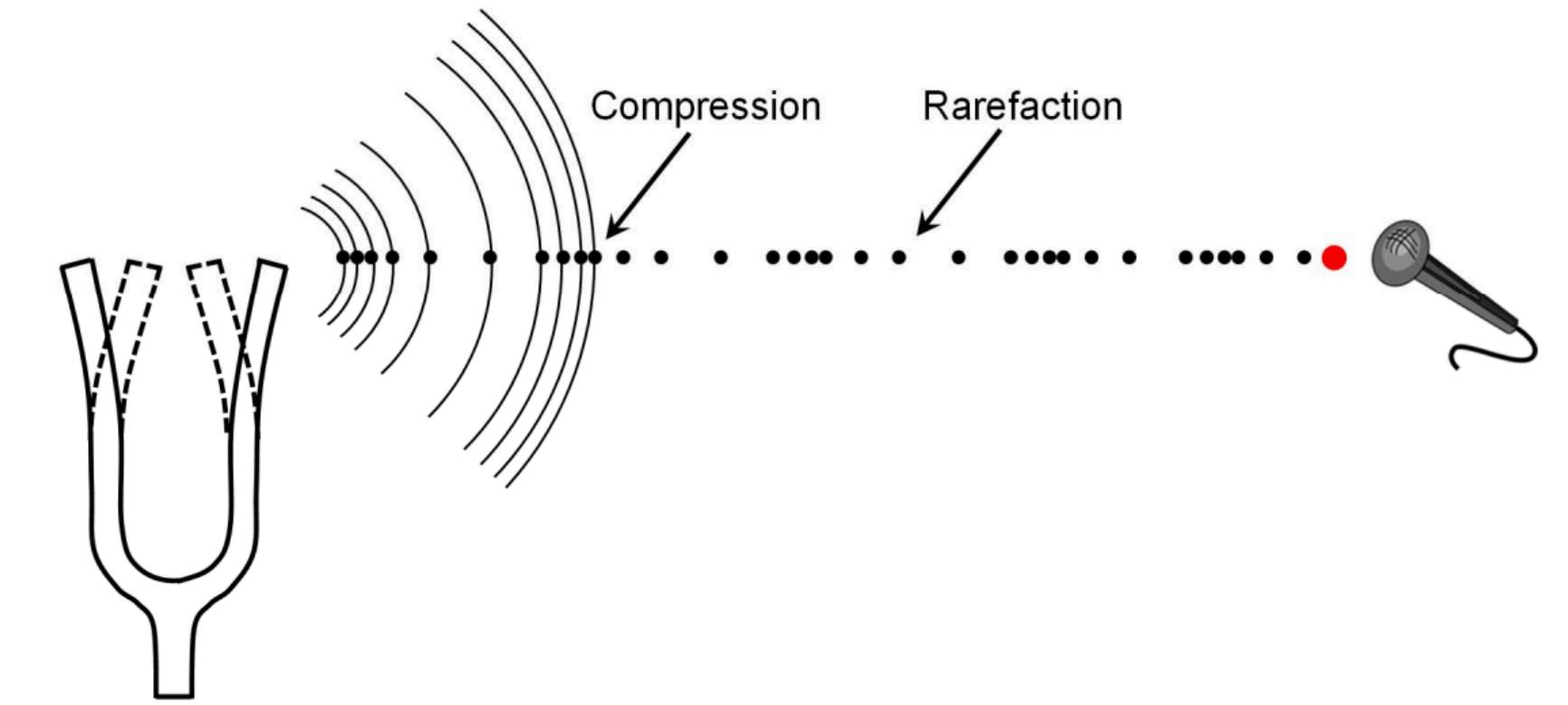


自然言語処理

1. 音響データの基礎

音響データ

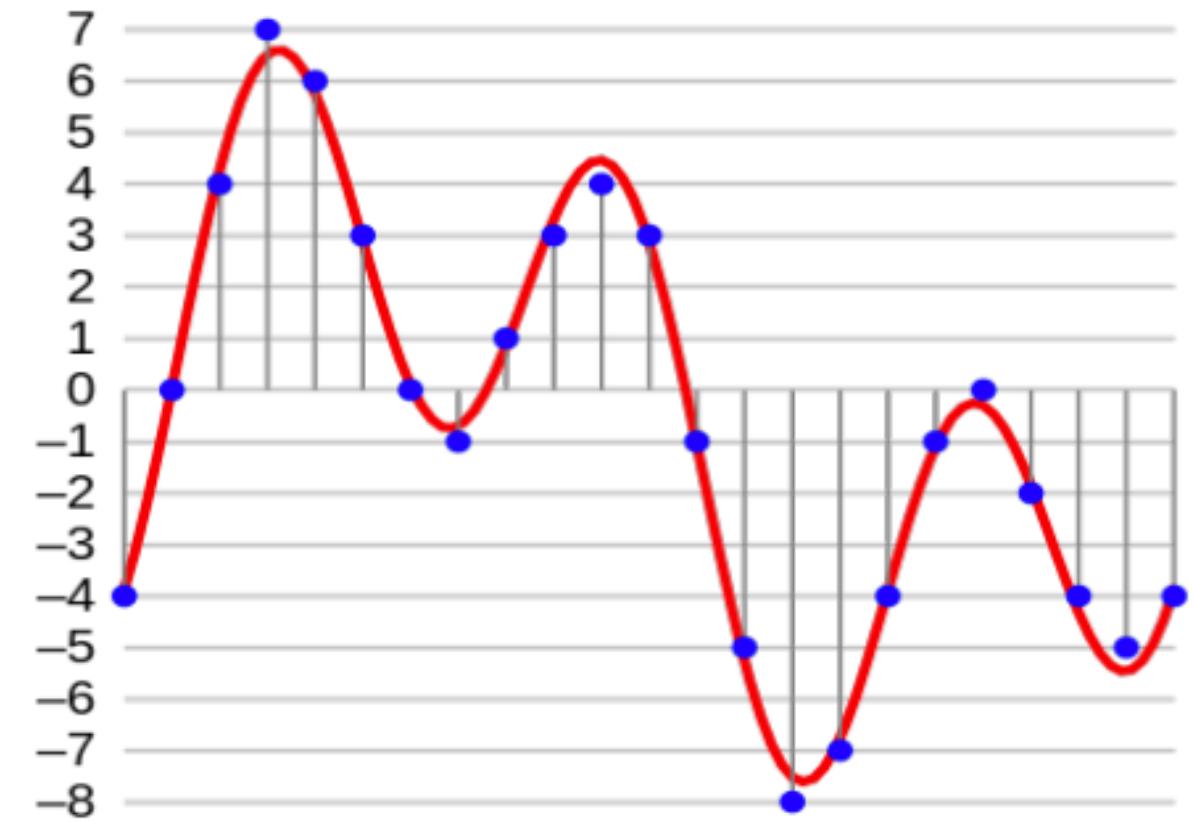
- 我々の身の回りの音は,
空気の振動 (を耳で感じ取って)
で生み出される
- 気圧の疎と密の振動的変化 = 音
- 気圧変化をプロットすると右の図
のようになる -> **波の形**
- 振幅が音の大きさ, 周期が音の高さに対応



音をコンピュータで扱う

7

- ・ サンプリングと量子化によって離散値に変換
- ・ サンプリング：値を一定時間ごとにサンプル（横）
 - ・ サンプリング定理：あるサイン波をサンプルする時は、間隔をその周期の半分よりも短く
- ・ 量子化：振幅を一定値ごとにサンプル（縦）
 - ・ 振幅値はバイナリ（0 or 1）で表現
 - ・ ビット数で表現できる上限が異なる（16ビット： $-2^{15} \sim 2^{15}$ まで）

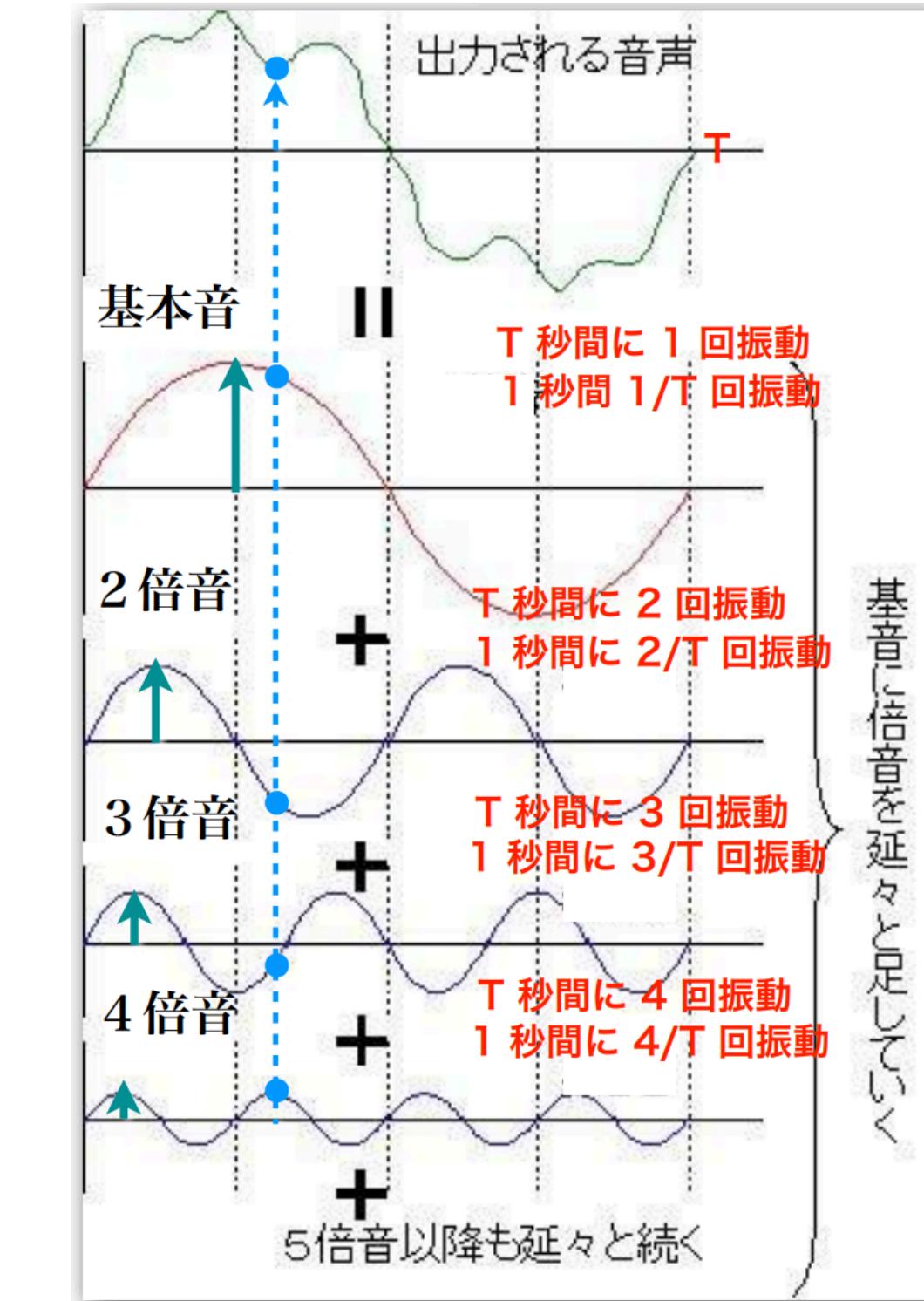


コンピュータ上では振幅・時間ともに
とびとびの離散信号として表現

音を解析する方法

- 実際の音はグチャグチャな波形
- -> 複数の異なる周波数の波 (sin波) の重ね合わせと考える
- コンピュータ上では離散フーリエ変換 (Discrete Fourier Transform; DFT) を適用して絶対値を取ると、周波数ごとの成分の強さを得ることができる

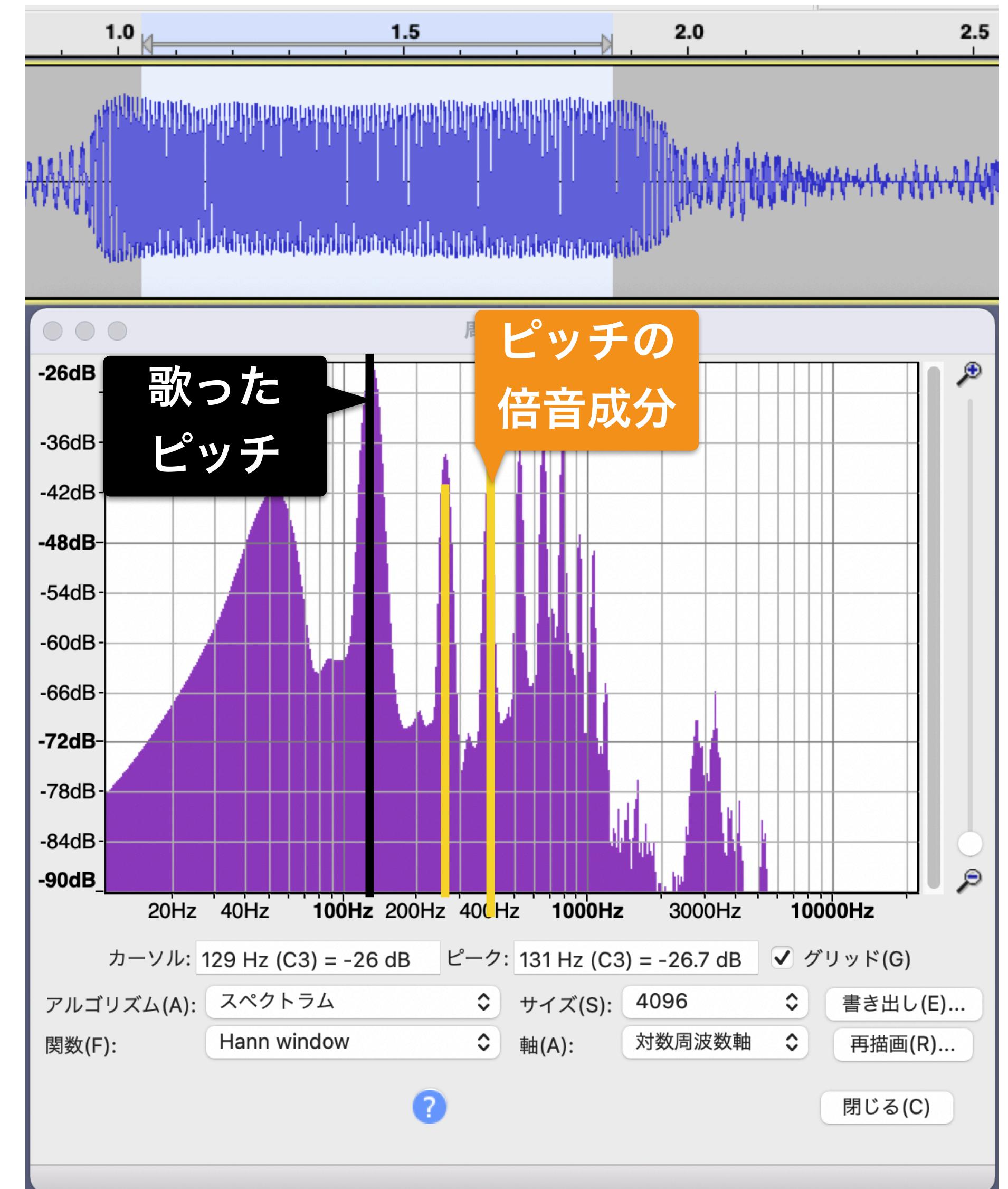
山本の「あ」
発声



フーリエ変換による周波数解析

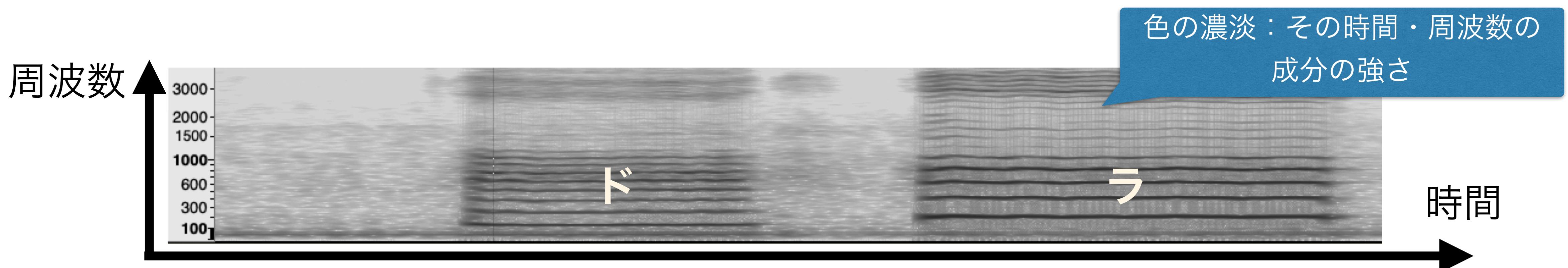
- 「あ」の声 (ド, C3)

- audacityで解析してみると131Hzの波が一番強い
- 倍音(2倍; 262Hz, 4倍; 524Hz)や、その他の高さの音の成分(n次倍音、非整数次倍音)も含む
- この成分比が音色の構成要素の一つ



- 音を時間×周波数の2次元の表現形式にしたもの

- 振幅スペクトログラム：短時間フーリエ変換（Short-Time Fourier Transform; STFT）を適用後、絶対値を取ることによって得られる。通常スペクトログラムと言わいたらこれ
- 非定常信号（時間的に変化する音）を解析したい場合に用いる

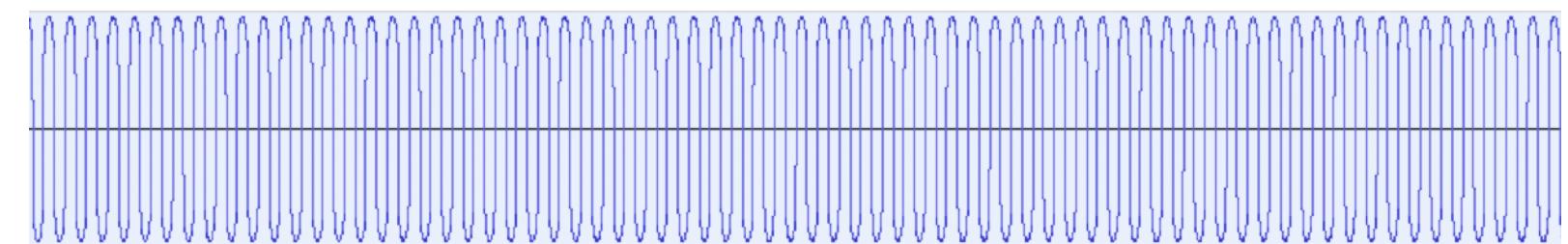


短時間フーリエ変換

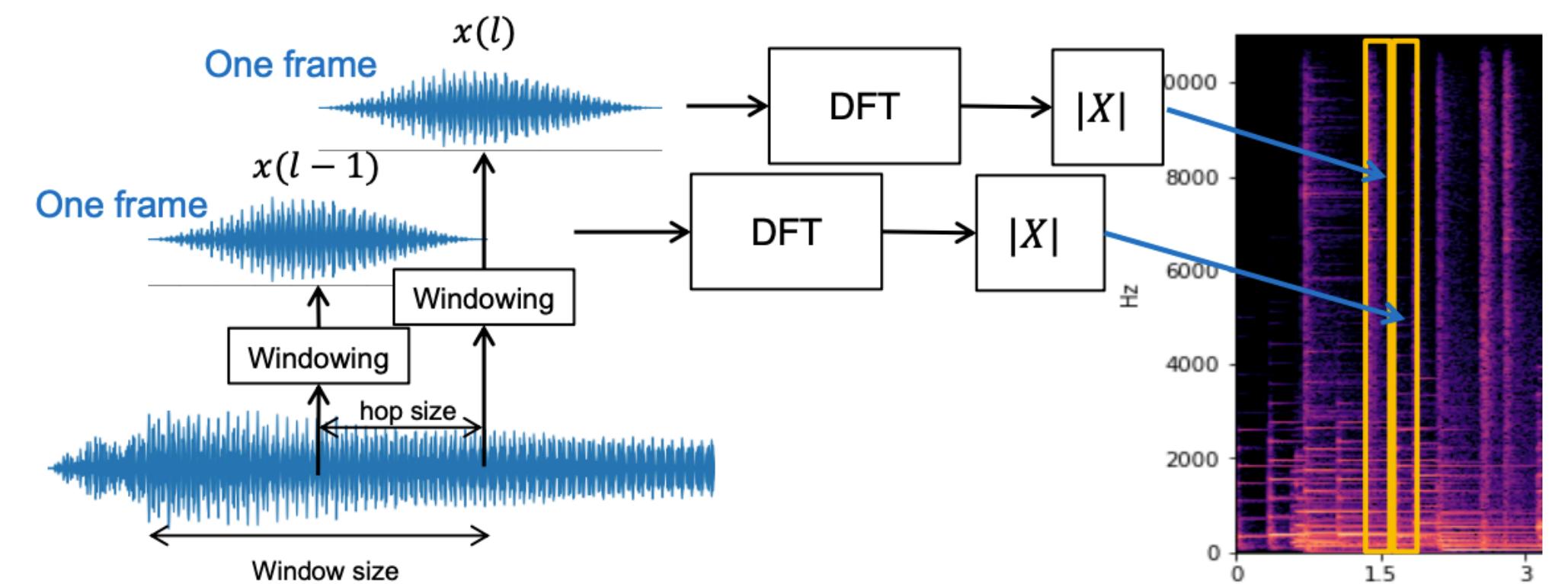
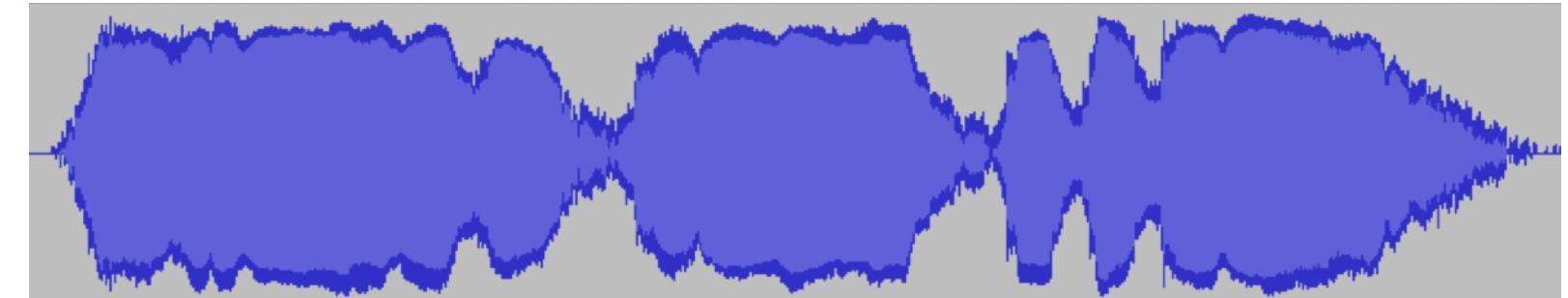
11

- 非定常信号はごく短い時間では定常であると仮定して、信号を短く区切ってフーリエ変換を行う処理
- 切り出した1単位をフレームと呼ぶ
- フレームは窓関数をかけた状態で切り出す
 - Window size: 窓関数の長さ
 - hop size: 次のフレームとのずらし幅

定常信号→全体でフーリエ変換



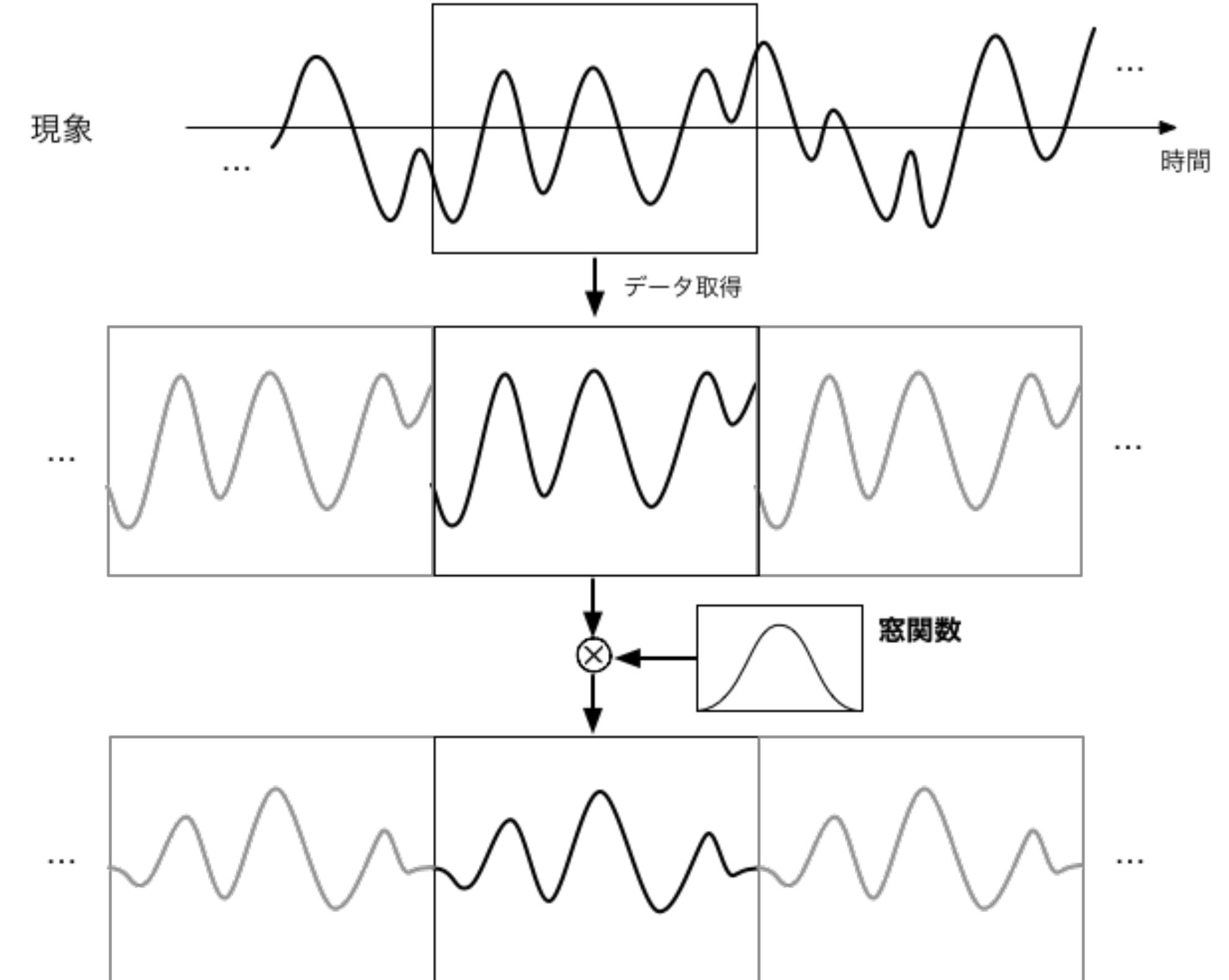
非定常信号→一刻一刻の周波数成分を知りたい



元音源（生波形）

スペクトログラム

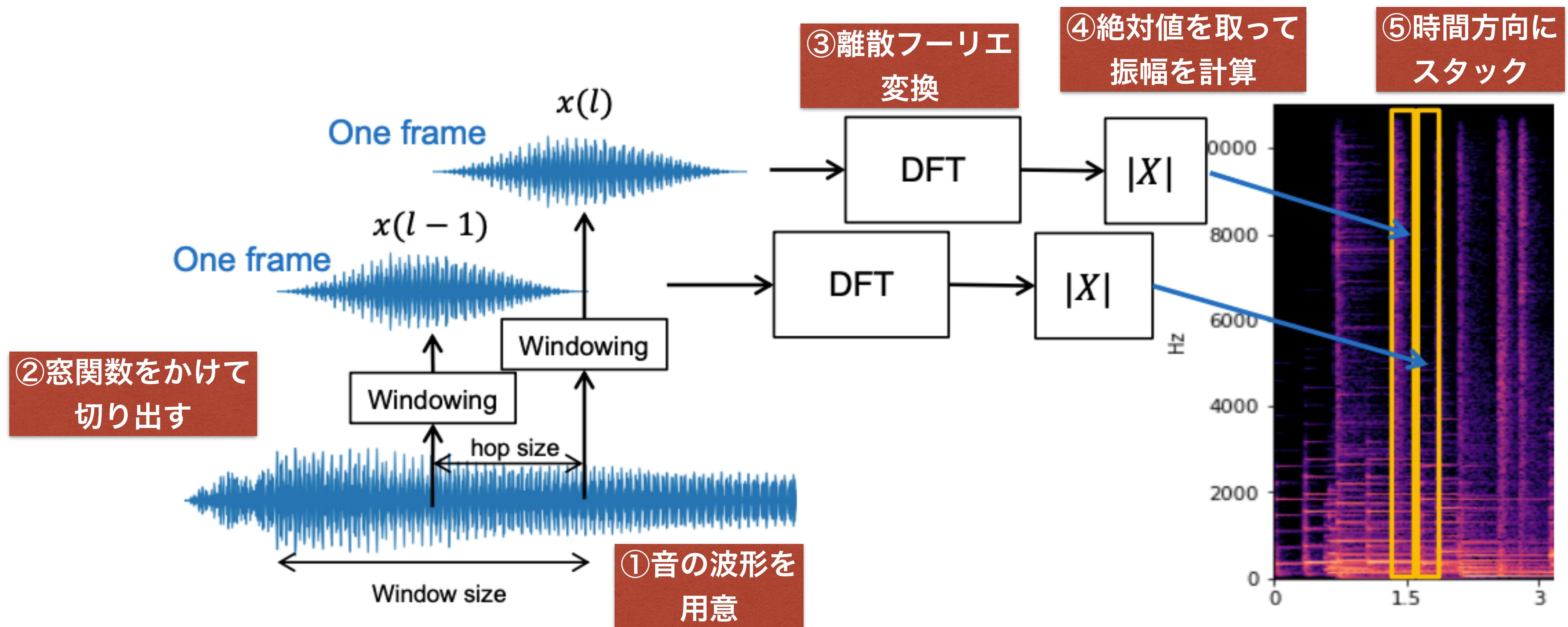
- 短時間フーリエ変換を行う際にはフレームに
対しフーリエ変換を適用する
- 端点に**不連続点**が生じてしまい悪影響が出る
 - 本来ない周波数成分が出現してしまう
- 窓関数**をかけて不連続な端点をおさえて計算
する



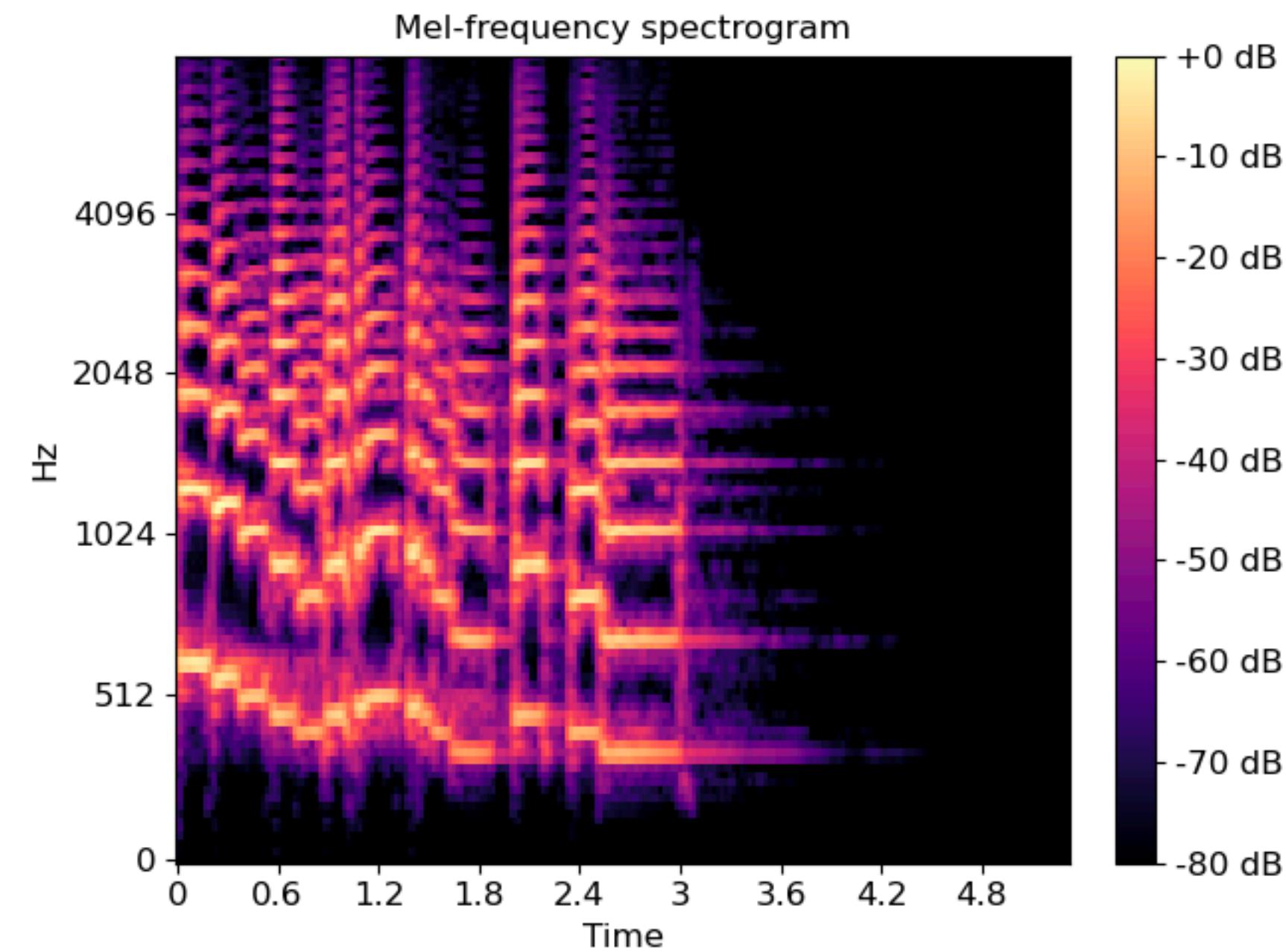
[https://wagtail.cds.tohoku.ac.jp/coda/python/
p-7-array-time-series-periodicity.html](https://wagtail.cds.tohoku.ac.jp/coda/python/p-7-array-time-series-periodicity.html)

全体像

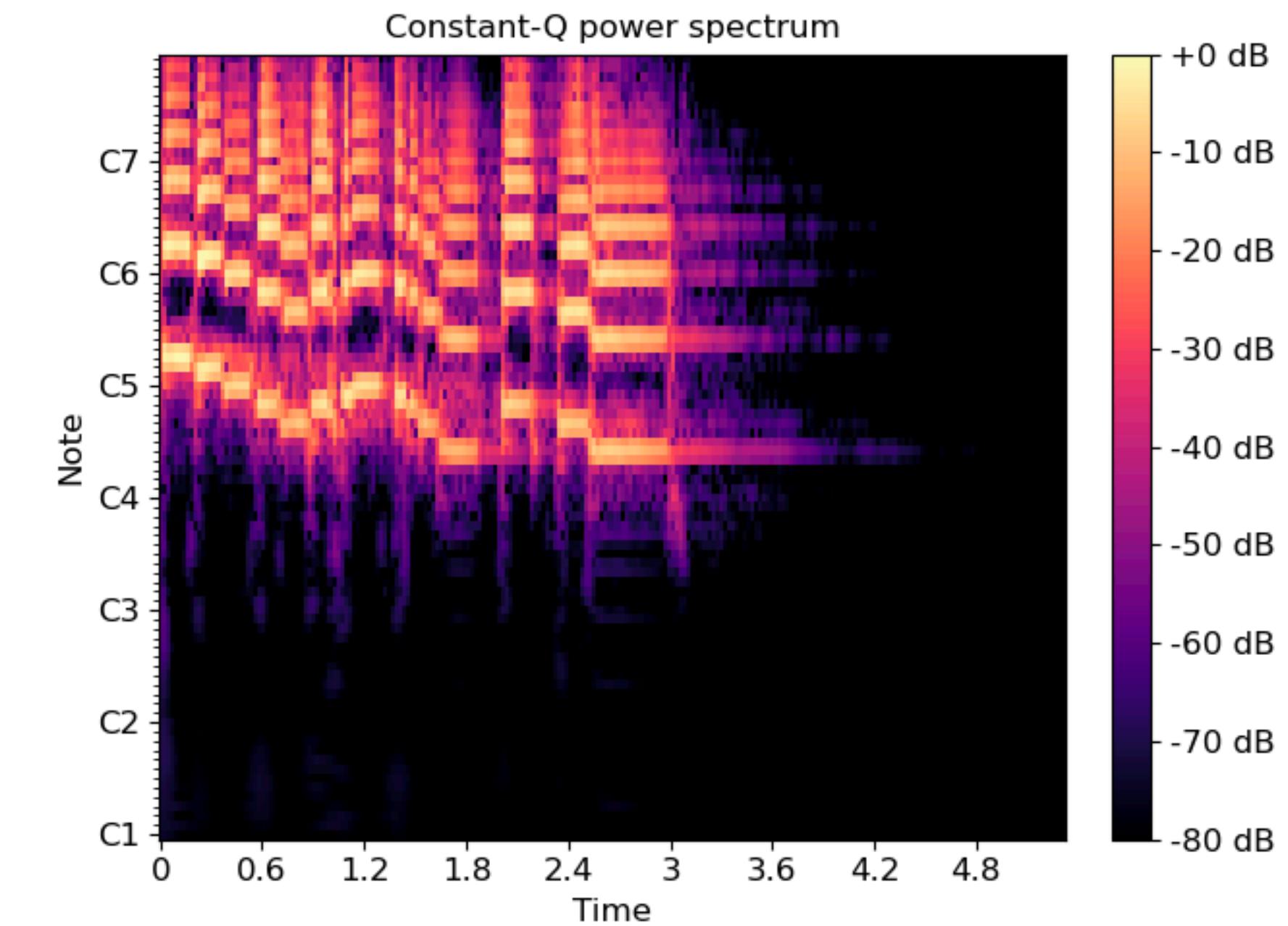
13



メルスペクトログラム



CQTスペクトログラム



人間の聴覚の特性を反映（低音の音高変化には敏感、
高音には鈍感）して
周波数軸のスケールを変換（メルフィルタバンクを適用）
音楽だけではなく、音声・環境音等にもよく使われる

Constant-Q Transformを適用すると得られるスペクトログラム。
低域の周波数解像度が高く得られる、各ビン（スペクトログラム
行列の行）が音高に対応（ビン/オクターブ=12の場合）
自動採譜など、音高が重要なタスクではこっちも使われる

おすすめライブラリ

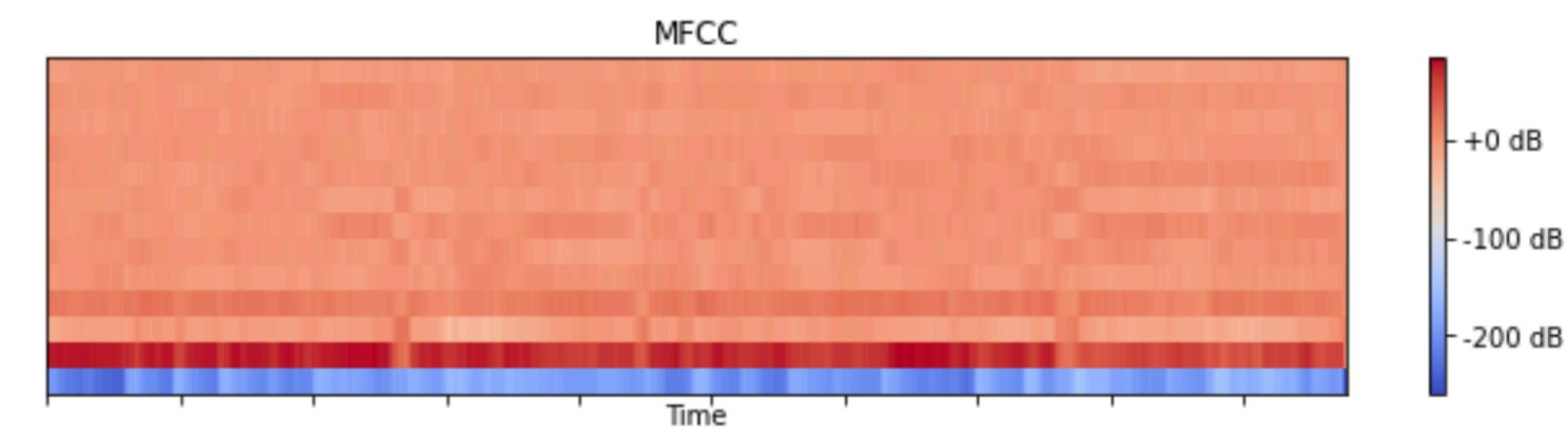
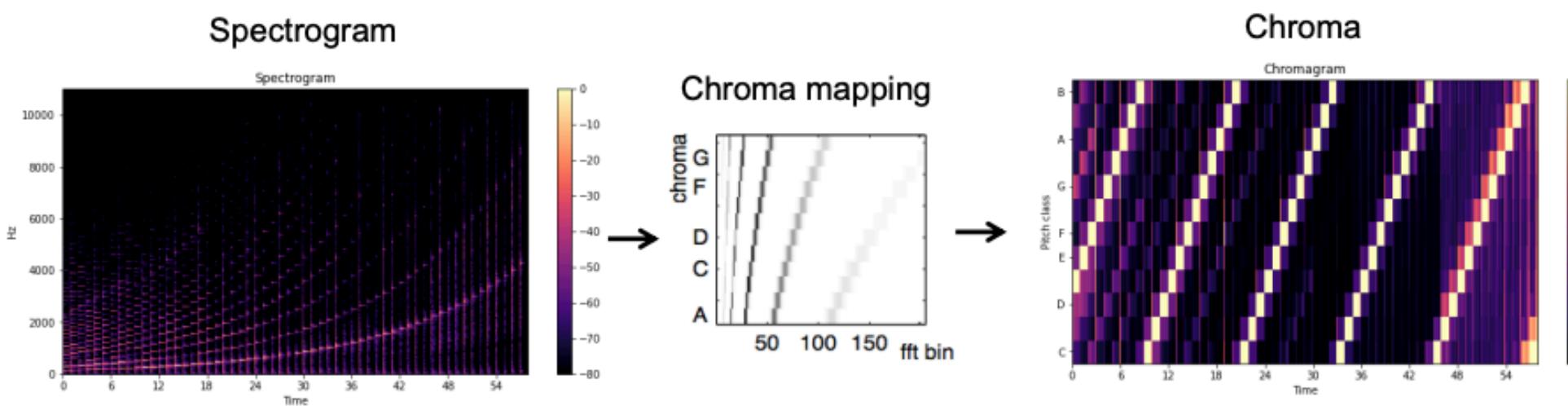
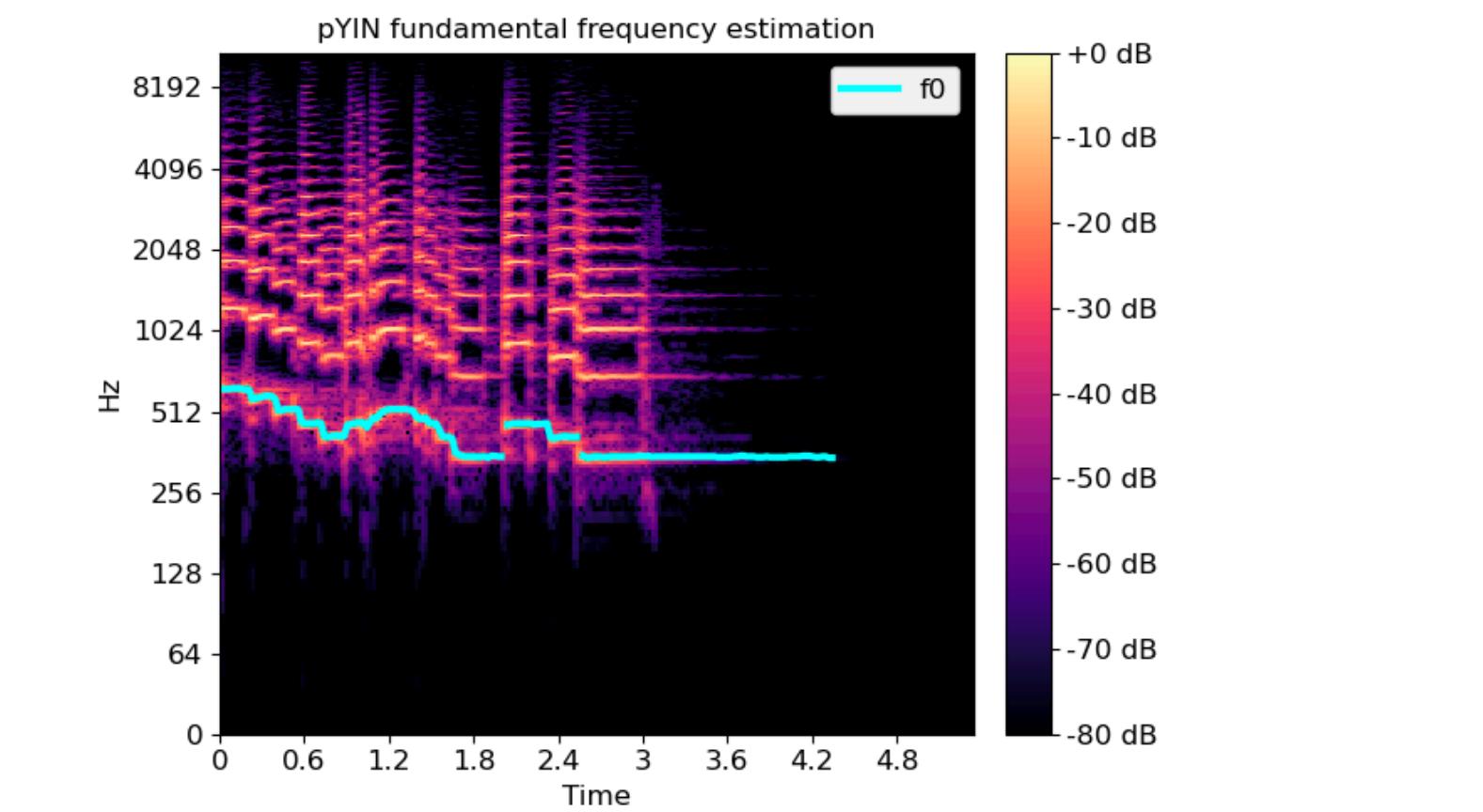
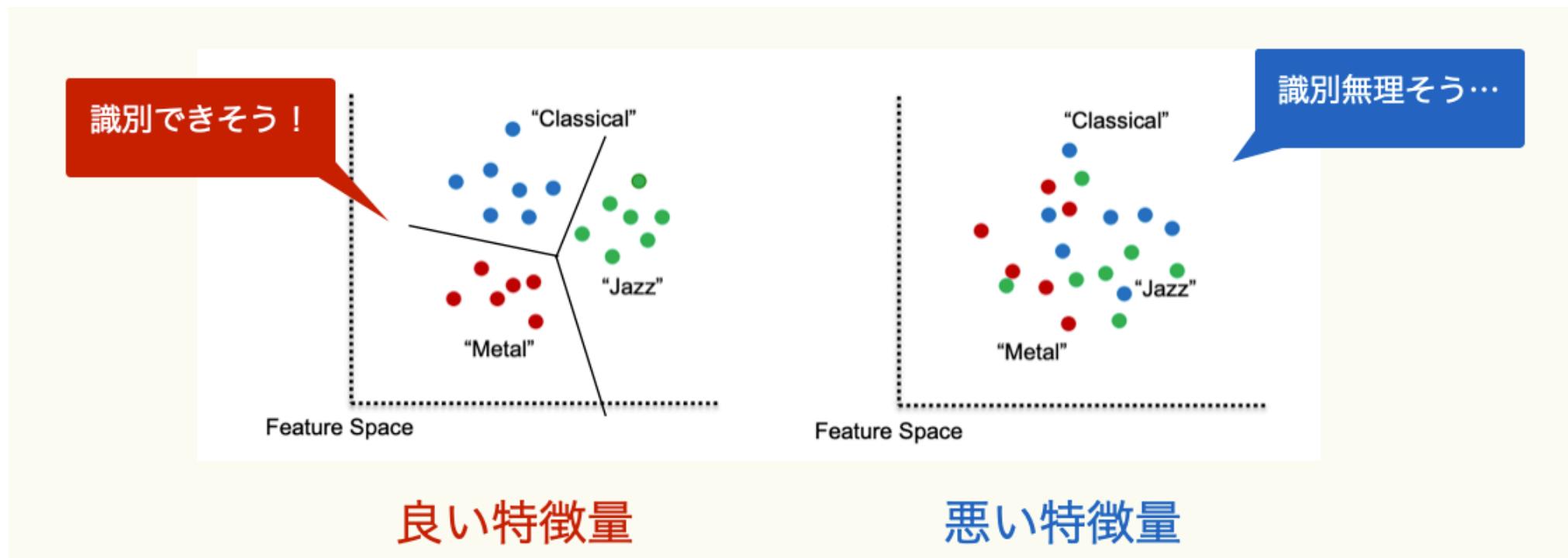
15

- Librosa
 - 神
- pysptk
 - 音声処理をやるならこっちもあり
- torchaudio
 - GPUが使える場合に.

次回予告：音響特徴量

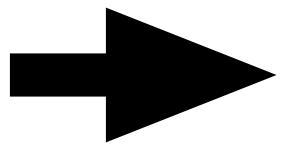
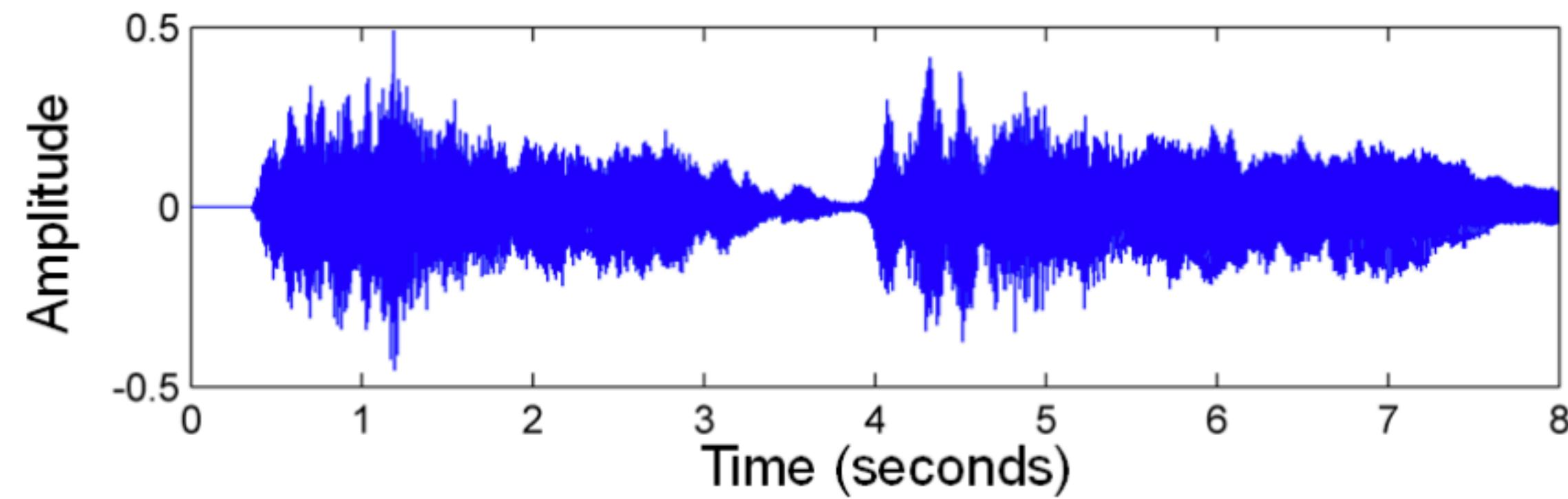
16

音響データを特徴量に変換する話は次回やります



2. 楽譜データ

- 音楽演奏を符号化したもの



Allegro con brio ($\text{♩} = 108$)

Allegro con brio ($\text{♩} = 108$)

ff

1 2

Red. *

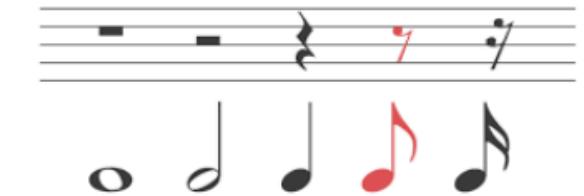
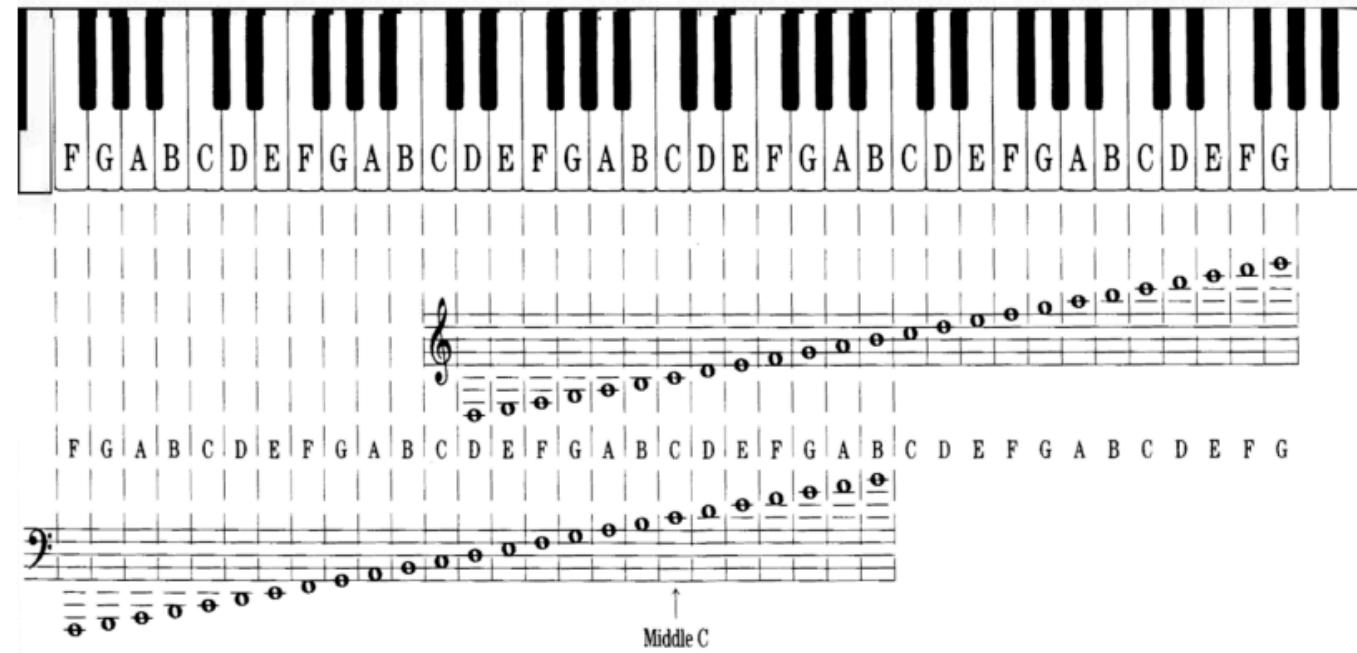
Red. *

楽譜・音符に含まれる要素

19

- 音符そのもの
 - 音高
 - ピッチクラス : C, D, E, ...
 - オクターブ : 4, 5, ... -> C4, E5 というように表現
 - ノートナンバーで表現されることも
 - 音長
 - 音価 : 全, 二分, 四分...
 - (テンポとセットでないと実際の秒数はわからない)

- Note: pitch, duration, velocity



{*ppp, pp, p, f, ff, fff*}

ノートナンバー

	4	3	5	4
C	60	261.6	まんなかのド	
C#	61	277.2		
D	62	293.7		
D#	63	311.1		
E	64	329.6		
F	65	349.2		
F#	66	370.0		
G	67	392.0		
G#	68	415.3		
A	69	440.0	時報の音	
A#	70	466.2		
B	71	493.9		
C	72	523.3		
C#5	73	554.4		
D	74	587.3		
D#	75	622.3		
E	76	659.3		
F	77	698.5		
F#	78	740.0		
G	79	784.0		
G#	80	830.6		
A	81	880.0		
A#	82	932.3		
B	83	987.8		

http://www.asahi-net.or.jp/~hb9t-ktd/music/Japan/Research/DTM/freq_map.html

データで指定されているものから、文脈から辿る情報も...

- リズム関連情報
 - ビート, 小節線, 拍子, テ
 - 調・コード
 - 調：長調, 短調等
 - コード進行
 - 演奏記号
 - 音の強さ：p, mp, mf, f, ...
 - アーティキュレーション：
スタッカート, ...
 - 発想記号：legato, marcato

MIDI



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

各音符のON/OFF情報を時系列にしたファイル

Music XML

```
<note>
  <pitch>
    <step>E</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>2</duration>
  <type>half</type>
</note>
```



各音符の情報をマークアップ形式で記述

ABC

```
X:1
T:sooranbushi
M:2/4
L:1/8
K:F
C2 DF | A2 GF | A2 GF | G2 FC | D2 FC | D2 D2 | z2 z2 ||
zG AA | GA AA | GA AA | GF D2 | zA, CA, | CD GF | zG AF | DC FD | D z CC |
DF A2 | A3 c | G F2 C | D2 D2 | D2 z2 ||
```

<https://ja.wikipedia.org/wiki/ABC%E8%A8%98%E8%AD%9C%E6%B3%95>

文字を並べる記法

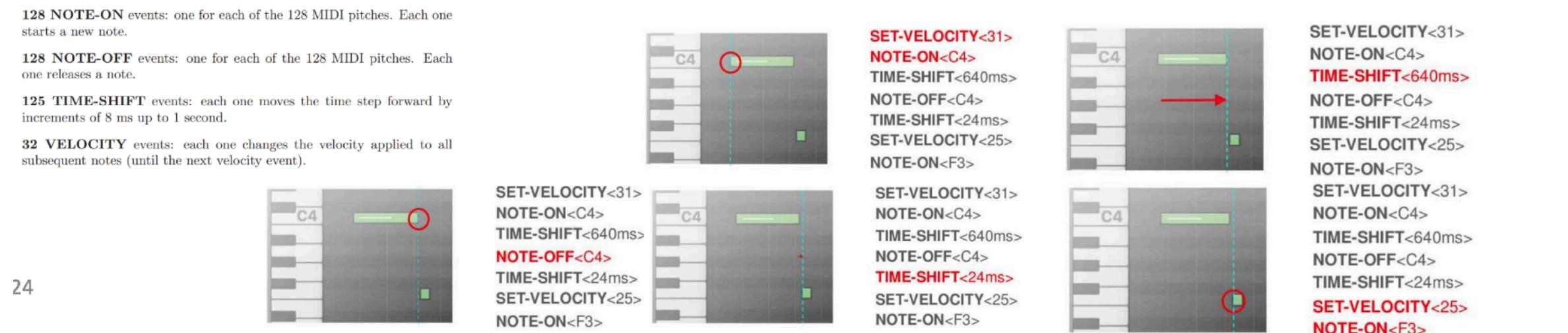
他にもあるけど滅多に見ないので割愛

特徴表現～スタンダードな方法～

22

MIDI記法

音高・音量・音長
ノートON/OFFのイベント列

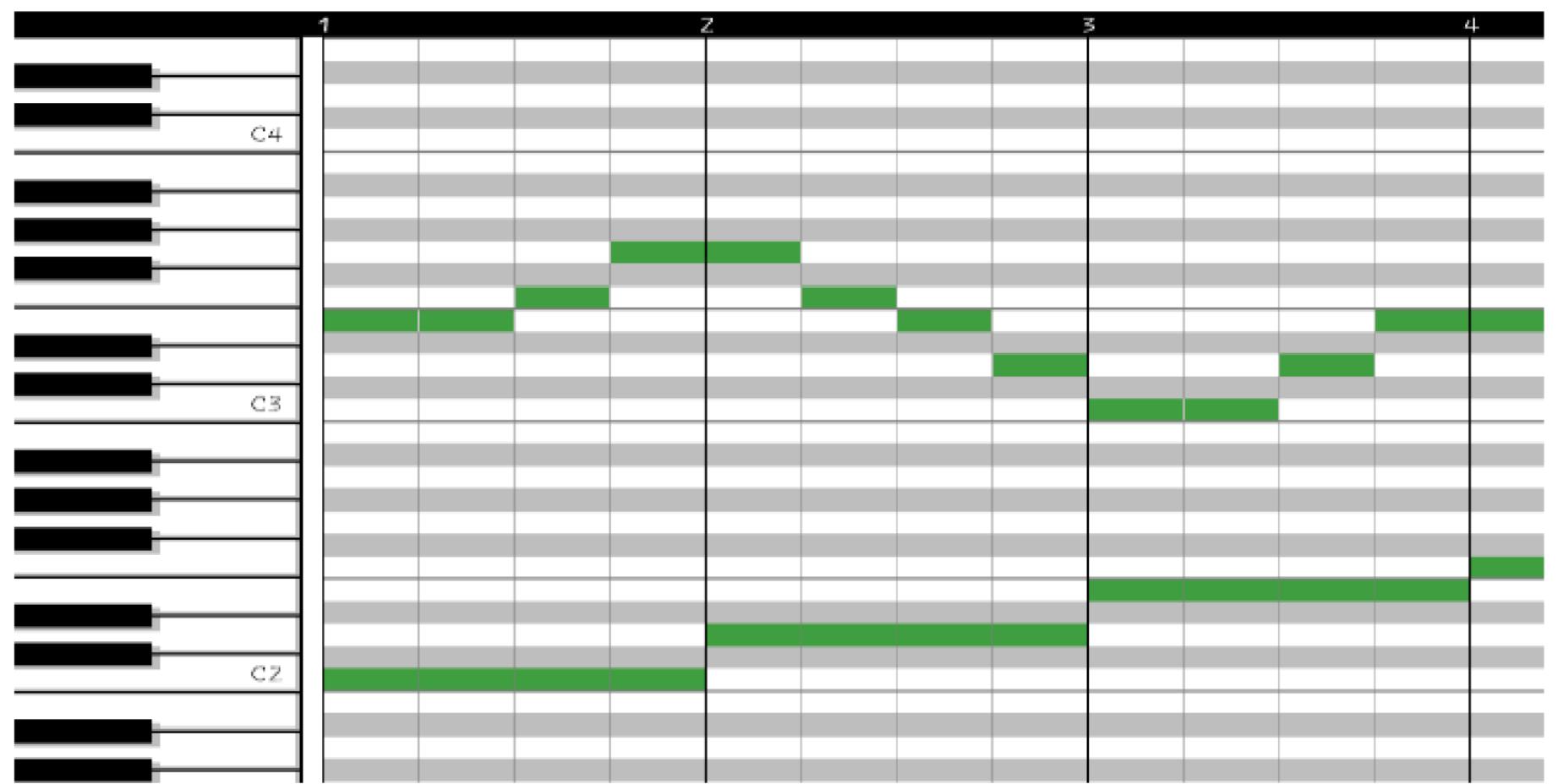


: 音符ごとの記述（処理で有用なことが多い）

: 拍の概念と相性×, 同時に多数の音が鳴る場合に効率が悪い, 音符の長さがわかりにくい

ピアノロール

音高×時間のバイナリ行列



: 直感的

: 密になる, 長い音なのか短い音の連発などの区別ができない（場合による）

トークン表現：音符の要素をシンボルとして並べる

REMI

全ての情報を順番に並べる

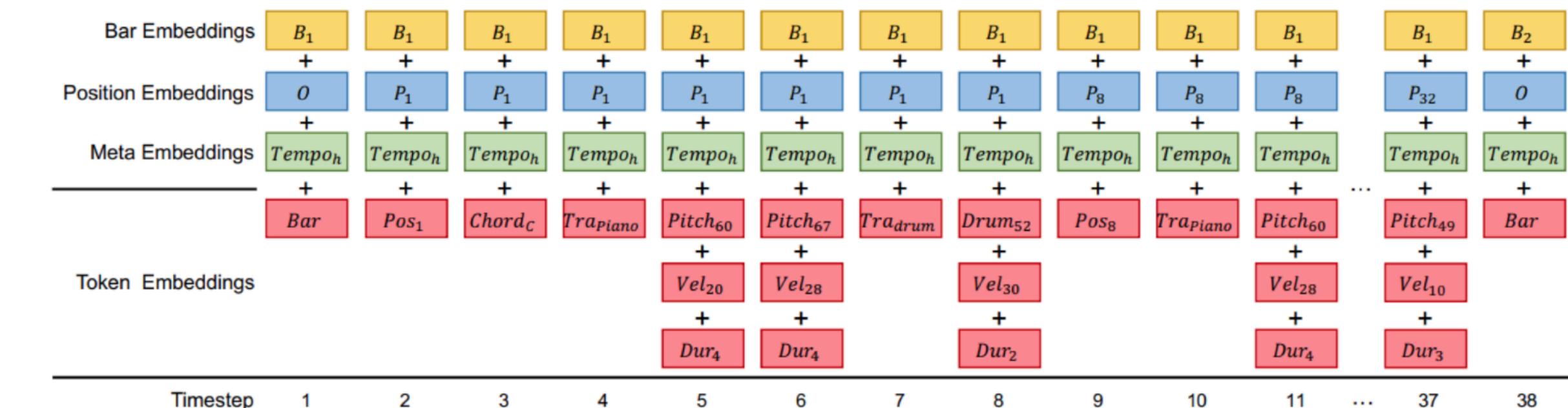


Bar, Position (1/16), Chord (C major),
Position (1/16), Tempo Class (mid),
Tempo Value (10), Position (1/16),
Note Velocity (16), Note On (60),
Note Duration (4), Position (5/16),
.....
Tempo Value (12), Position (9/16),
Note Velocity (14), Note On (67),
Note Duration (8), Bar

	MIDI-like [30]	REMI (this paper)
Note onset	NOTE-ON (0–127)	NOTE-ON (0–127)
Note offset	NOTE-OFF (0–127)	NOTE DURATION (32th note multiples; 1–64)
Time grid	TIME-SHIFT (10–1000ms)	POSITION (16 bins; 1–16) & BAR (1)
Tempo changes	x	TEMPO (30–209 BPM)
Chord	x	CHORD (60 types)

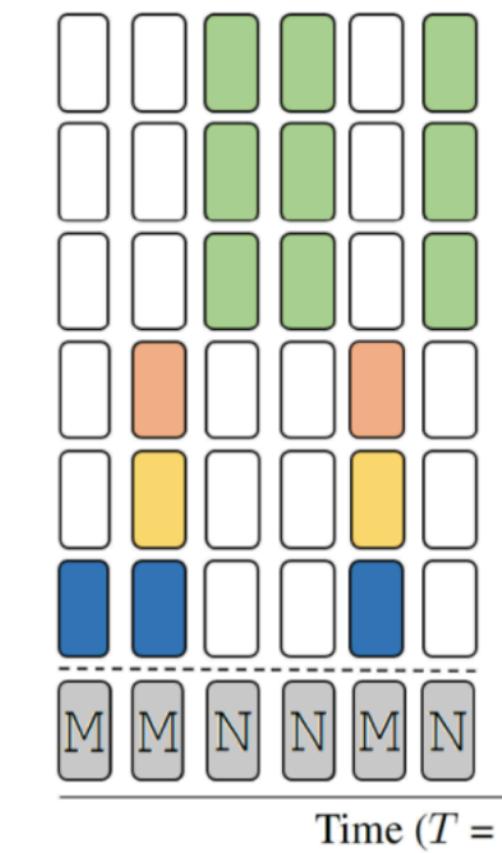
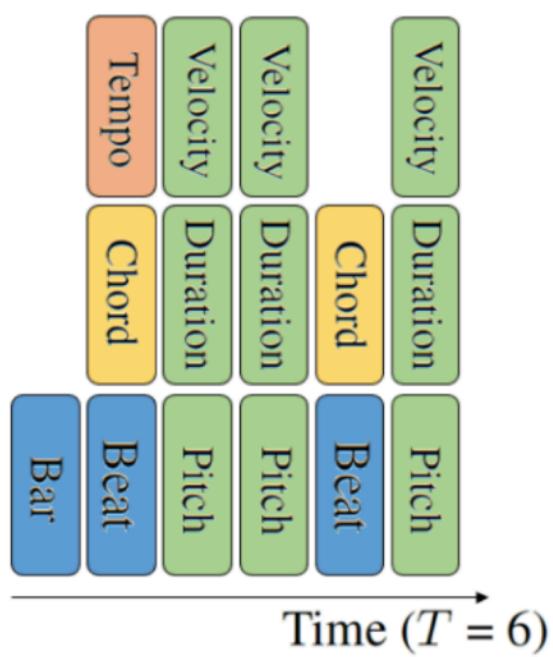
MuMIDI

マルチトラック対応



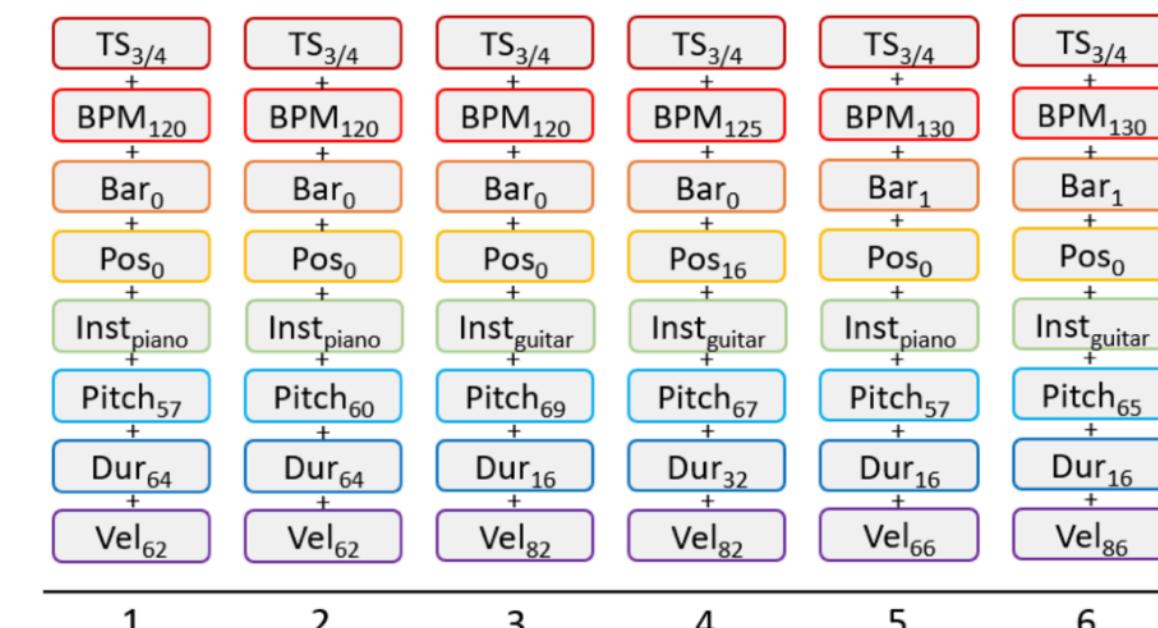
CP

多次元化



Octuple MIDI

多次元化・CPを
さらにコンパクトに



最近のTransformerベース音楽生成でよく用いられている。詳しくは [miditok](#) のページで

- MIDI : pretty-midi
- MusicXML : Music21
- トークン表現 : Miditok
- ピアノロールへの変換 : pypianoroll
- pythonでのデータハンドリング : Muspy

補足

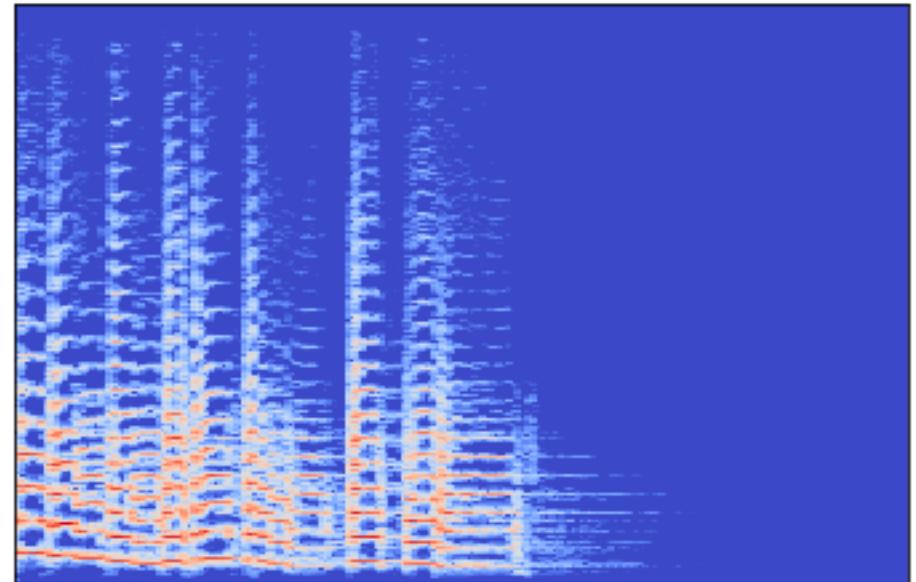
$$\begin{aligned}
 X(k) &= \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi kn}{N}} = \sum_{n=0}^{N-1} x(n)\left(\cos\frac{2\pi kn}{N} - j\sin\frac{2\pi kn}{N}\right) \\
 &\quad \boxed{e^{j\frac{2\pi kn}{N}} = \cos\frac{2\pi kn}{N} + j\sin\frac{2\pi kn}{N}} \quad \text{オイラーの公式} \\
 &= \sum_{n=0}^{N-1} x(n)\cos\frac{2\pi kn}{N} - j \sum_{n=0}^{N-1} x(n)\sin\frac{2\pi kn}{N} = X_{re}(k) + jX_{im}(k)
 \end{aligned}$$

$$X_{mag}(k) = \sqrt{X_{re}(k)^2 + X_{im}(k)^2} \quad X_{phase}(k) = \tan^{-1}\left(\frac{X_{im}(k)}{X_{re}(k)}\right)$$

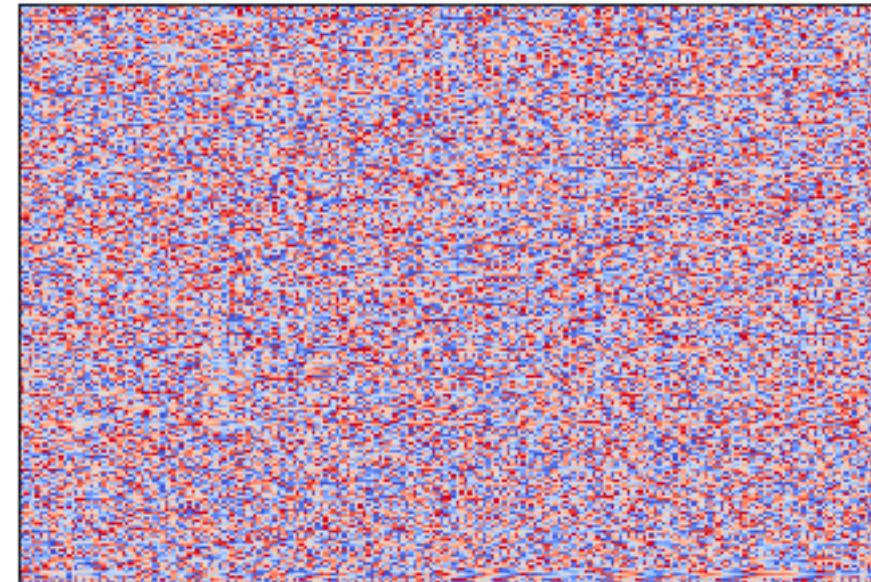
- ある離散信号 $x(n)$ と, k 個の異なる周波数成分をもつ, 長さ N の複素信号 $e^{j2\pi kn/N}$ の内積 (複素数で得られる)
- 絶対値で振幅, 偏角で位相が得られる

位相スペクトログラム

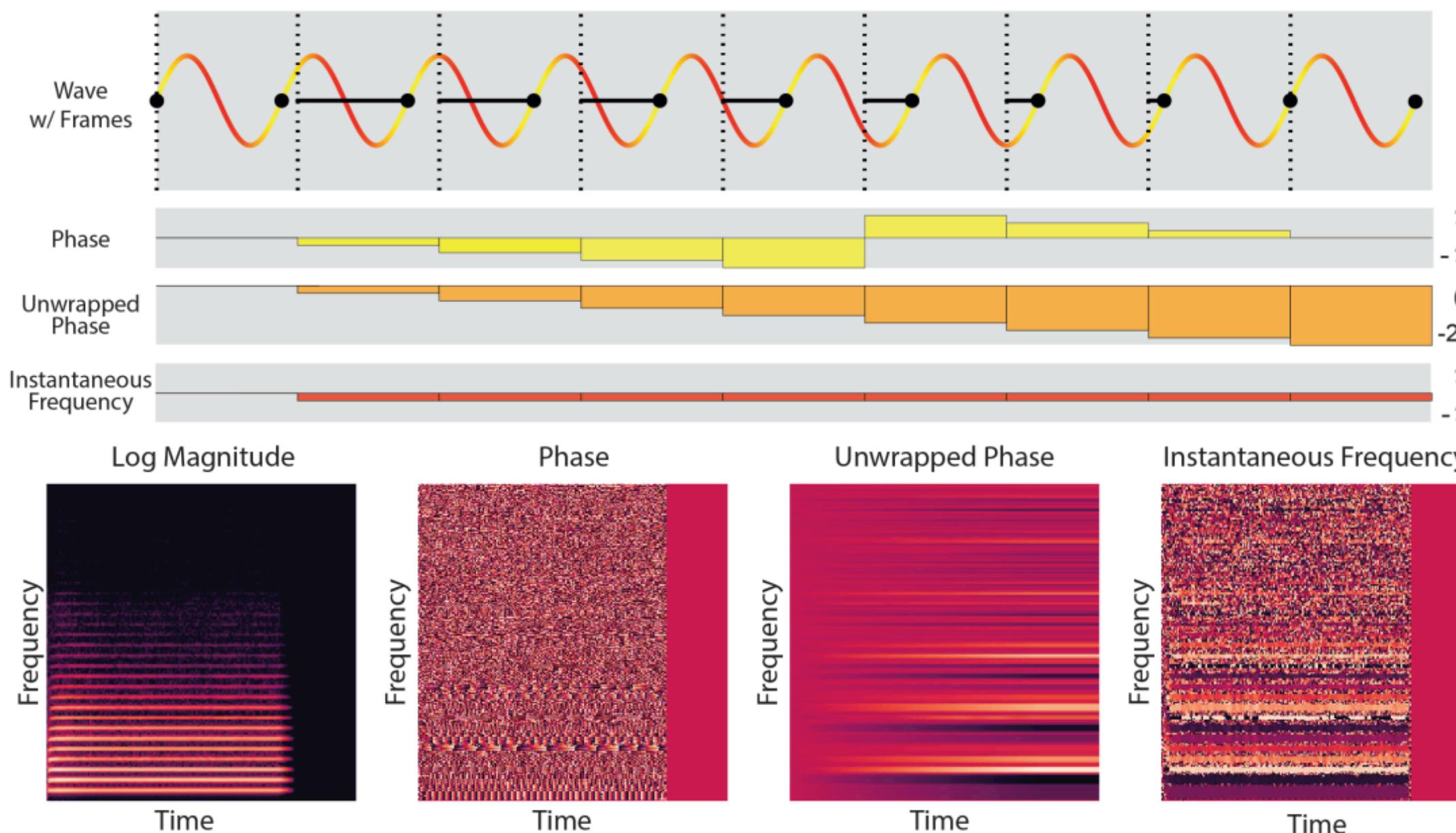
27



振幅



位相

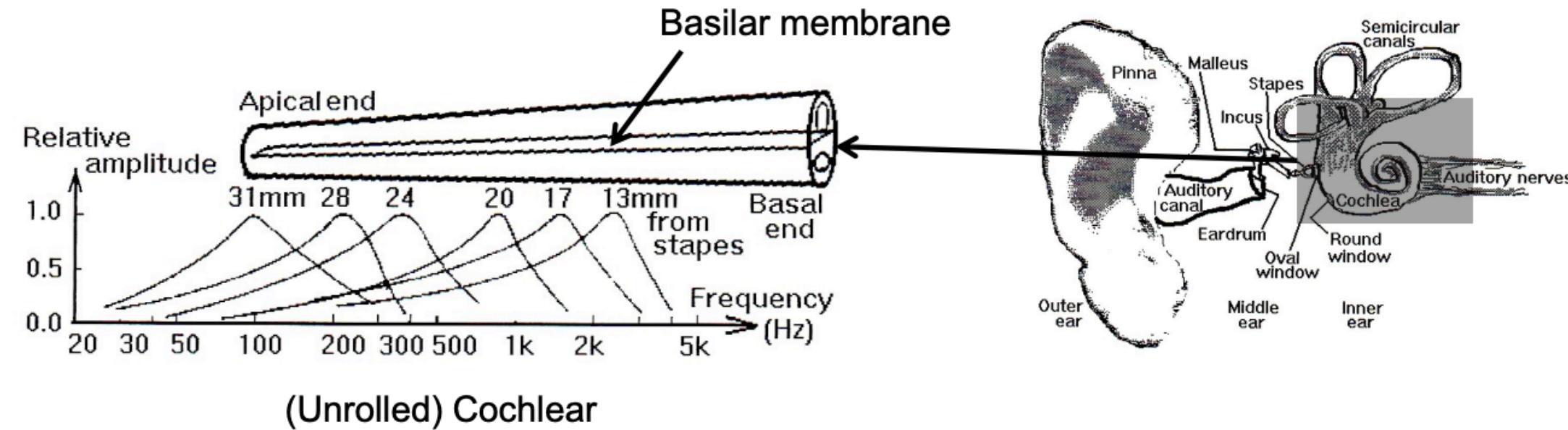


GANSynthでは位相も考慮

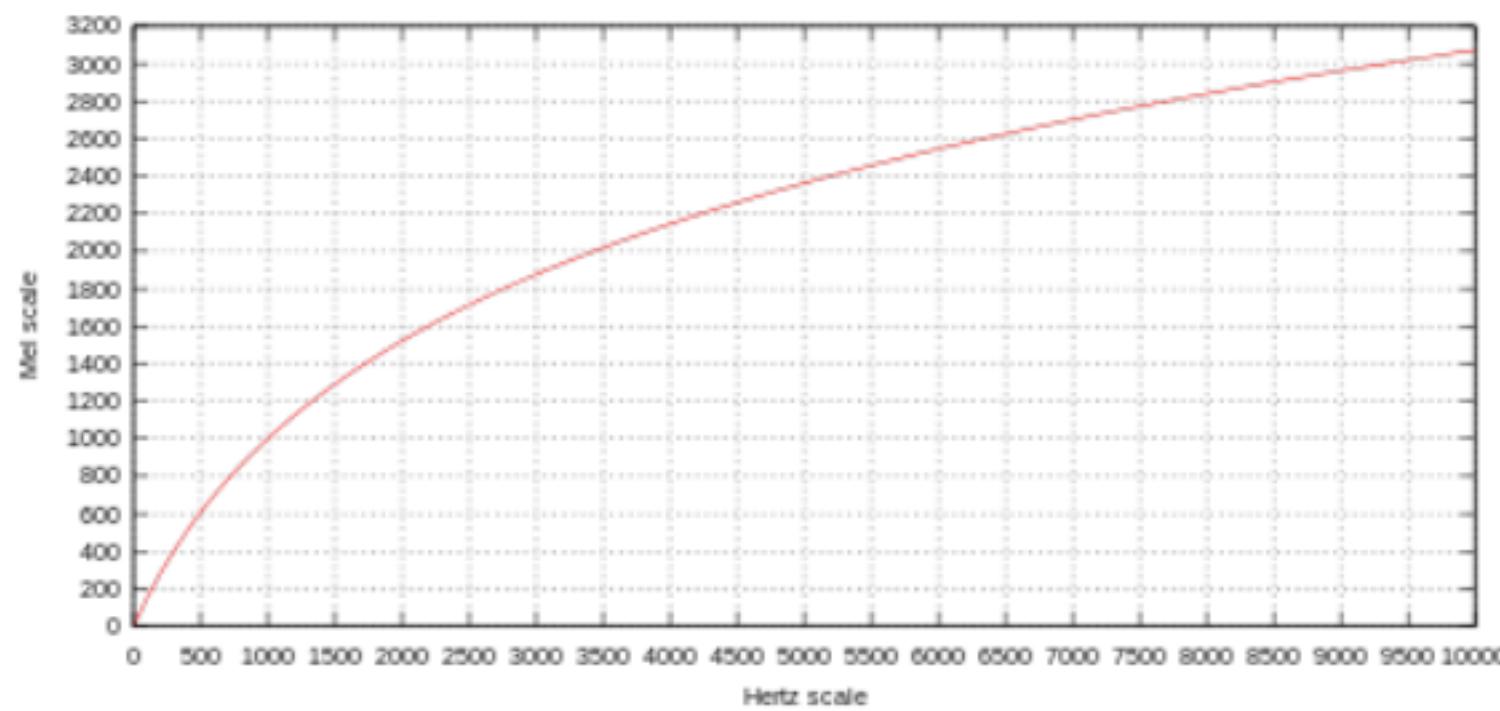
- STFTにより得られる位相スペクトログラム
- 振幅スペクトログラムから音を復元するには位相が必要（位相を+して逆STFTという操作をする）
- 音の合成においては位相が音の品質に影響を与えるため位相のモデリングも重要なとされている
- 興味が湧いたら https://www.jstage.jst.go.jp/article/essfr/15/1/15_25/_pdf/-char/ja

メルスペクトログラムとメルフィルタバンク

28

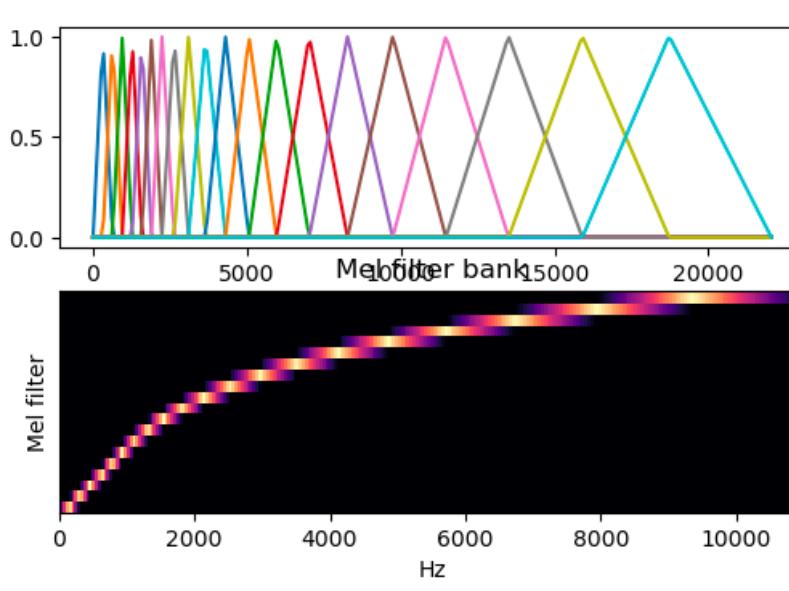
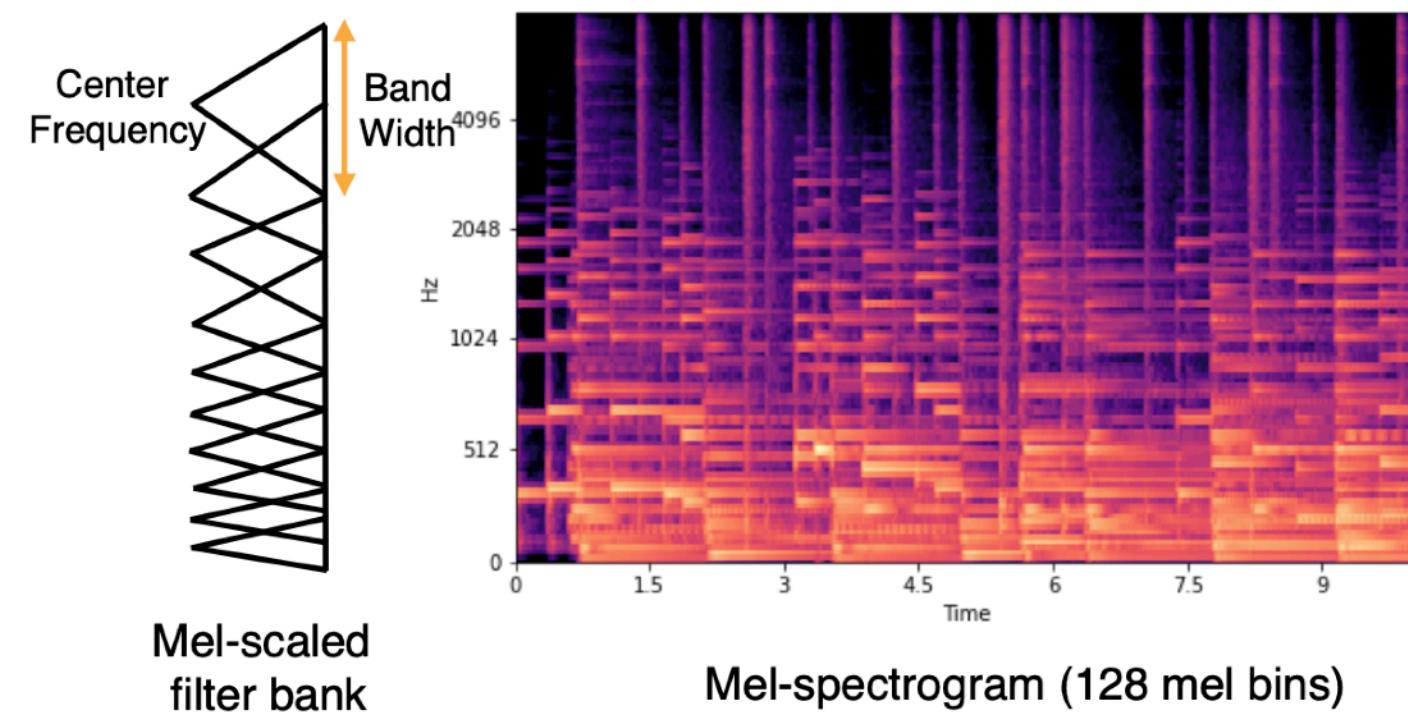


- 音を感じ取る器官の蝸牛は**バンドパスフィルタ**のような役割をしている（音の高さの弁別）

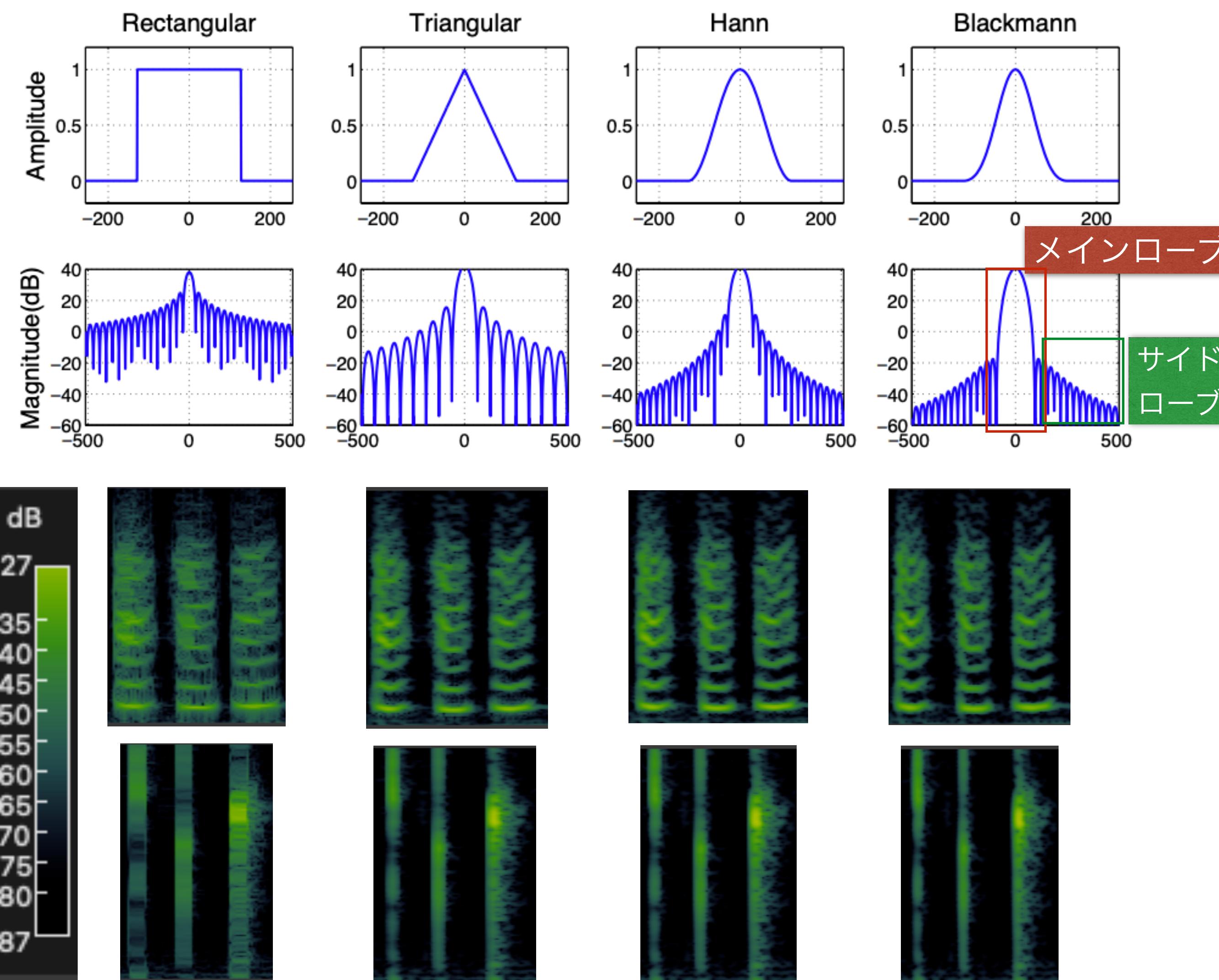


$$m = 2595 \log_{10}(1 + f / 700)$$

- 音の高さの**対数軸上で均等に並ぶ**=メル尺度
- 三角型の**バンドパスフィルタ**を↑に従って並べた**メルフィルタバンク**を、振幅スペクトログラムの各時間にかけ時間軸にスタックしたのが**メルスペクトログラム**



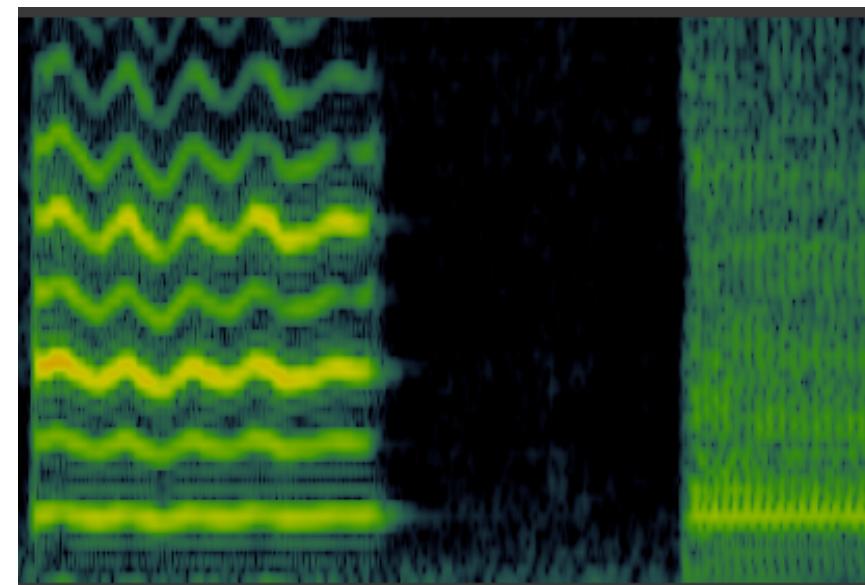
窓関数について



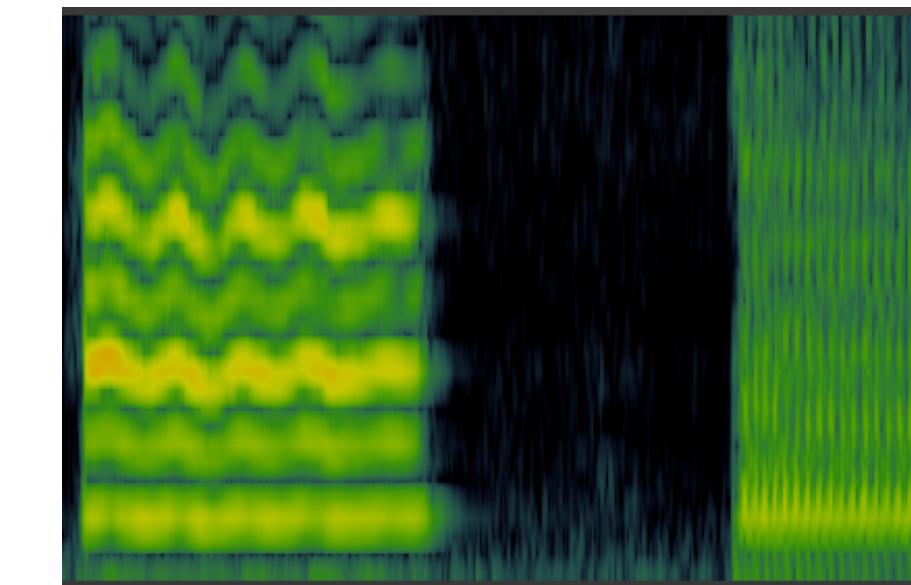
- 窓関数には種類がたくさん
 - “良い”スペクトログラムを作るには、**音の真の周波数成分を捉えているか？**が重要
 - 窓関数をかけた音の振幅スペクトルは、周辺の周波数帯にも成分が“染み出す”
- 染み出し具合=窓関数の性能
 1. メインローブの幅：狭いほど真の周波数の値がはっきりする
 2. サイドローブの高さ：低いほど周辺の周波数の振幅を抑えられる
 - ↑両者はトレードオフ
 - 音楽系だとHann窓がよく用いられる

- 窓関数の切り出す長さによって変化
 - 短く切り出す -> 周波数解像度：低 時間解像度：**高**
 - 長く切り出す -> 周波数解像度：**高** 時間解像度：低

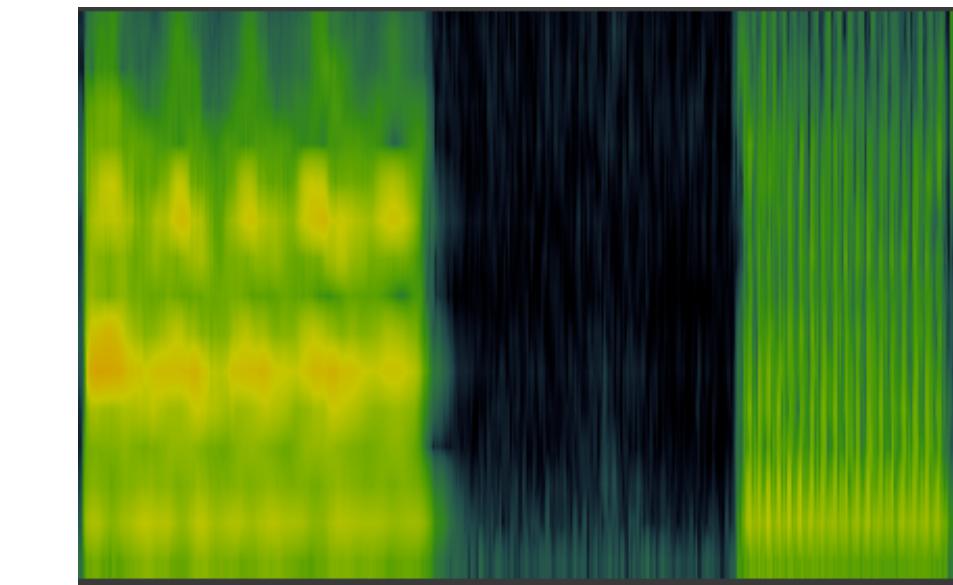
ex. ビブラートとリップトリル



2048サンプル



1024サンプル



512サンプル

山本のAPSIPA論文 (<https://www.slis.tsukuba.ac.jp/lspc/0000890.pdf>) では、異なる解像度のスペクトログラムを3つ用いることにより歌唱テクニックの認識性能を改善