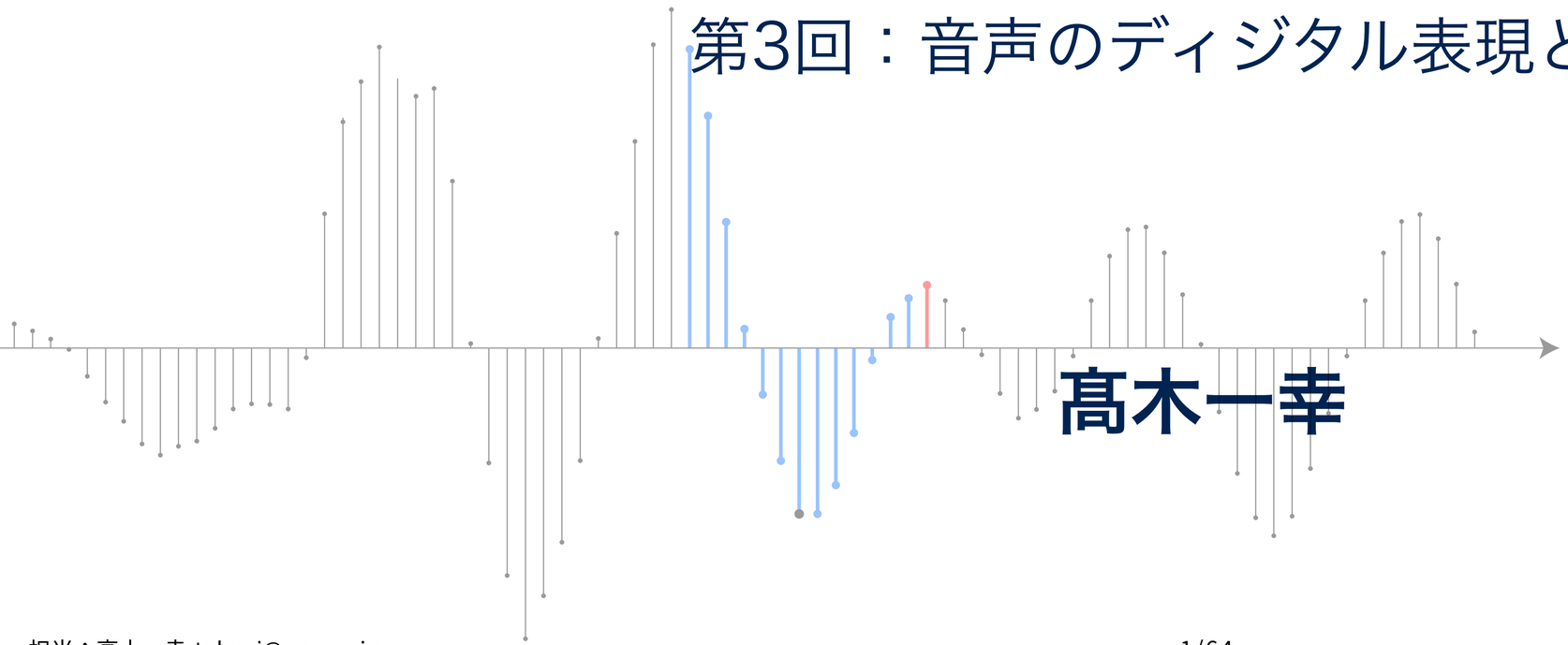
http://www.takagi.inf.uec.ac.jp/mmip/

マルチメディア処理

Multimedia Information Processing

第3回：音声のディジタル表現と処理



高木一幸

担当：高木一幸 takagi@uec.ac.jp

講義概要

- マルチメディアデータ（文書、音声、画像、映像など）の表現方法と処理技術について、基礎的な内容を紹介・解説する。
- 授業時間中および宿題として、計算機を使った演習を行うことにより実際のデータの扱い方を学び、理解を深める。

第1回：授業の概要説明、
人間の感覚とマルチメディア処理（4/12）

髙木

第2回：文字・テキストの表現と処理（4/19）

第3回：音声のディジタル表現と処理（4/26）

第4回：画像・映像のディジタル表現（5/10）

第5回：マルチメディアデータの符号化とファイル形式（5/17）

第6回：2次元図形の表現と描画（5/24）

第7回：画像処理(1)画素ごとの濃淡変換（5/31）

第8回：画像処理(2)空間フィルタリング（6/7）

廣田

第9回：カメラと写真撮影（6/14）

第10回：3次元コンピュータグラフィックス（6/21）
(1)形状表現と透視投影

第11回：3次元コンピュータグラフィックス（6/28）
(2)照明効果とシェーディング

第12回：アニメーションと映像制作（7/5）

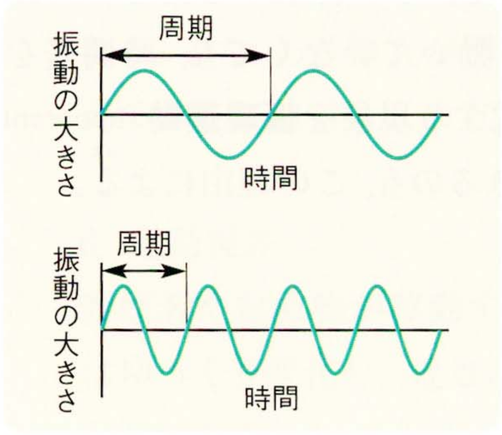
第13回：シミュレーションと可視化（7/12）

第14回：グラフィックパイプラインとシェーダ（7/19）

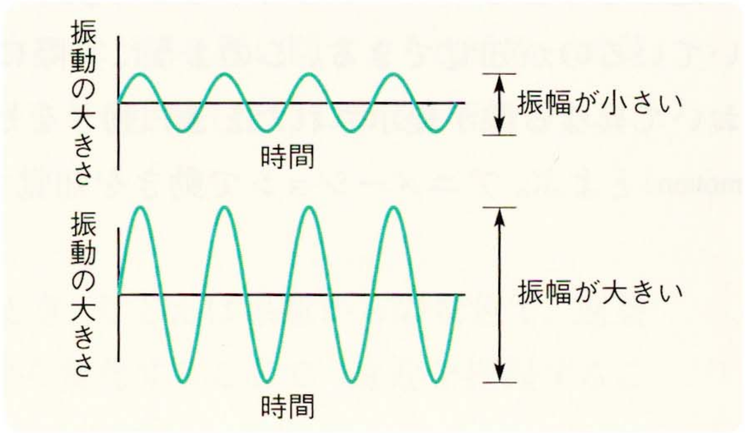
第15回：マルチメディアの応用例I（7/26）

担当：高木一幸 takagi@uec.ac.jp

音の構成要素



■図 1.16——音の高さ

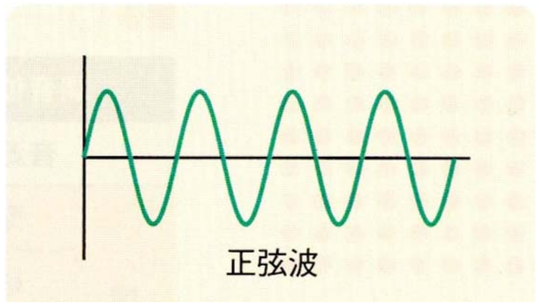


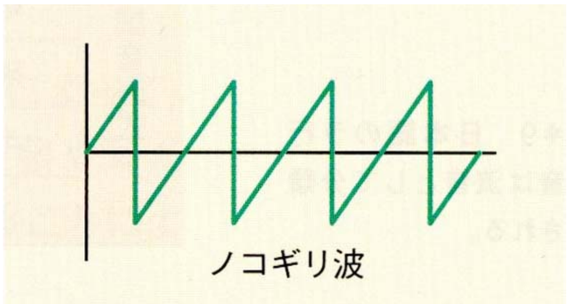
■図 1.17——音の大きさ

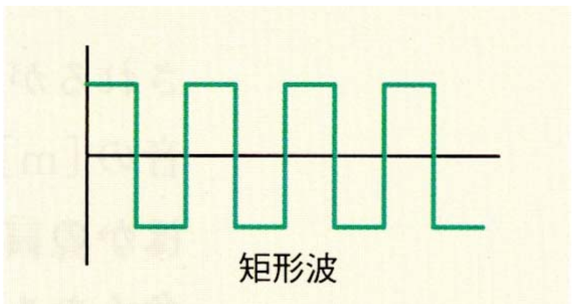
- **音の高さ（高低）pitch**：波形の周期の逆数である周波数 [Hz]。
  - 周波数が大きいほど高い音。
  - 人間の可聴周波数は約20〜20,000Hz。
- **音の強さ（音量）volume, loudness**：波形の振動の大きさ。
  - **マスキング効果 masking effect**：ある音が別の大きな音によって妨害され聞こえなくなる。
  - **カクテルパーティー効果 cocktail party effect**：大勢の人の声が騒がしいパーティー会場でも意識を向けた会話のみを聞き分けることができる。

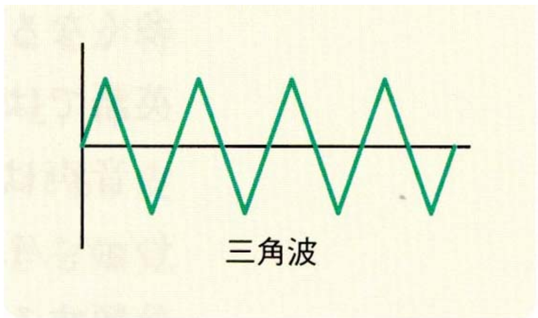
音の構成要素

- **音色 tone**：波形の形状による
  - 波形に含まれる周波数成分が異なると異なった音色に聞こえる。









■図 1.18——さまざまな音の波形

音の構成要素

- 音の印象 auditory impression

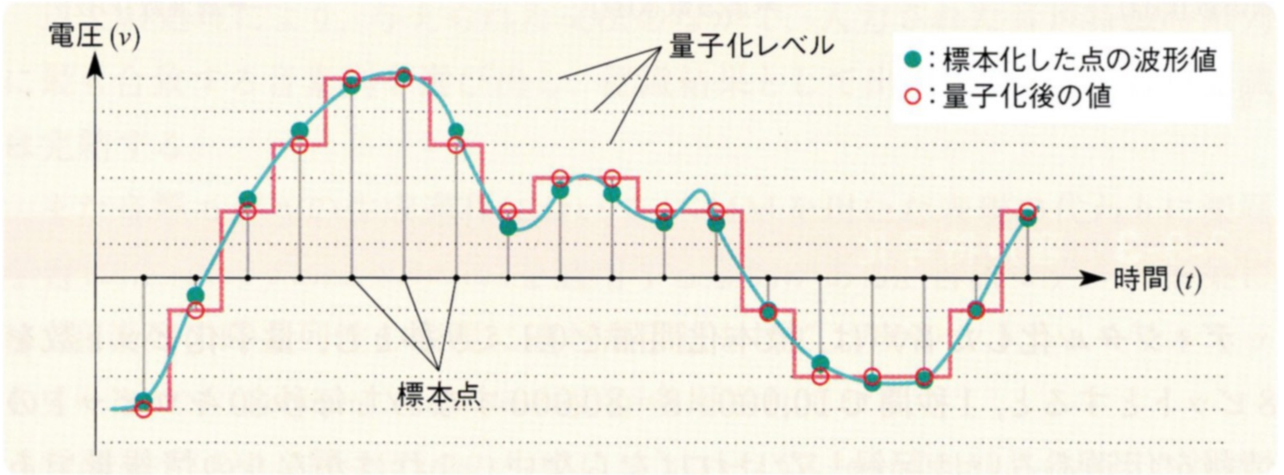
  - 音を構成する要素が変わることにより、音の印象も変化する。
  - 音の物理的な性質により、異なった心理効果がもたらされる。

■表 1.2——音の物理的性質と印象

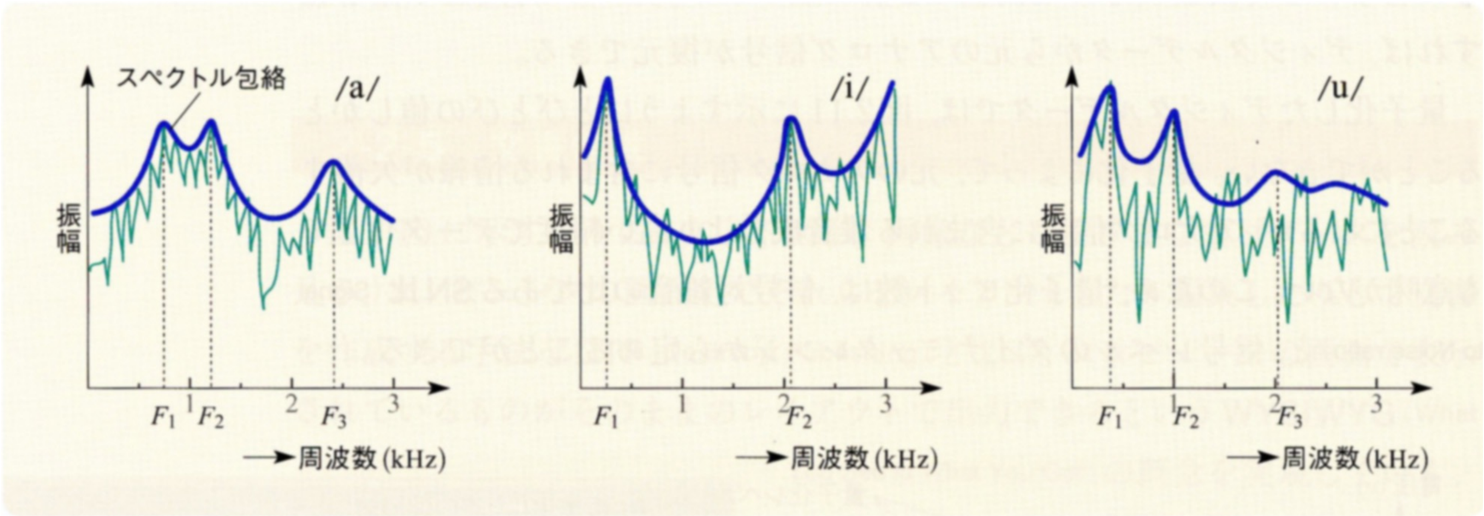
音の物理的性質	音色の印象	関係する印象語対
音の高低	金属的性質	鋭い－鈍い，高い－低い，鮮やか－ぼけた， 硬い－柔らかい，派手な－地味な
音の高低差 （音程，和音）	美的性質	澄んだ－濁った，滑らかな－ざらざらした， 乾いた－潤いのある，張りのある－しわがれた
音の大きさ	力動的性質	迫力のある－物足りない，力強い－弱々しい， 重い－軽い，大きい－小さい

音声のディジタル表現と処理

1. 音の標本化と量子化
2. 音声符号化
3. 音声波形
4. 音声認識
5. 音声合成



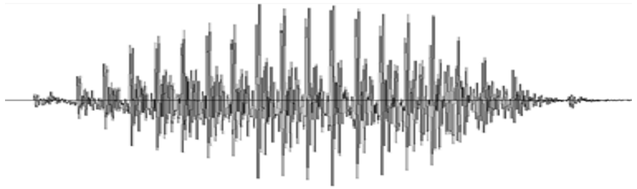
■図 2.11——信号レベルの量子化



■図 2.13——母音スペクトルの例

音の標本化と量子化

A/D変換・D/A変換



Analog信号

コンサートホール音楽

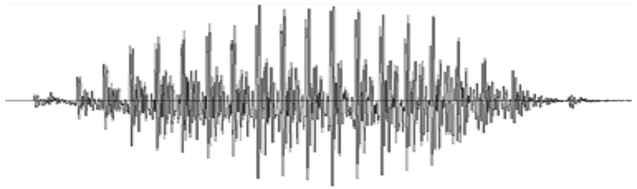
人間の会話

Analog to **D**igital 変換

Digital 処理

分析、蓄積、変換、合成

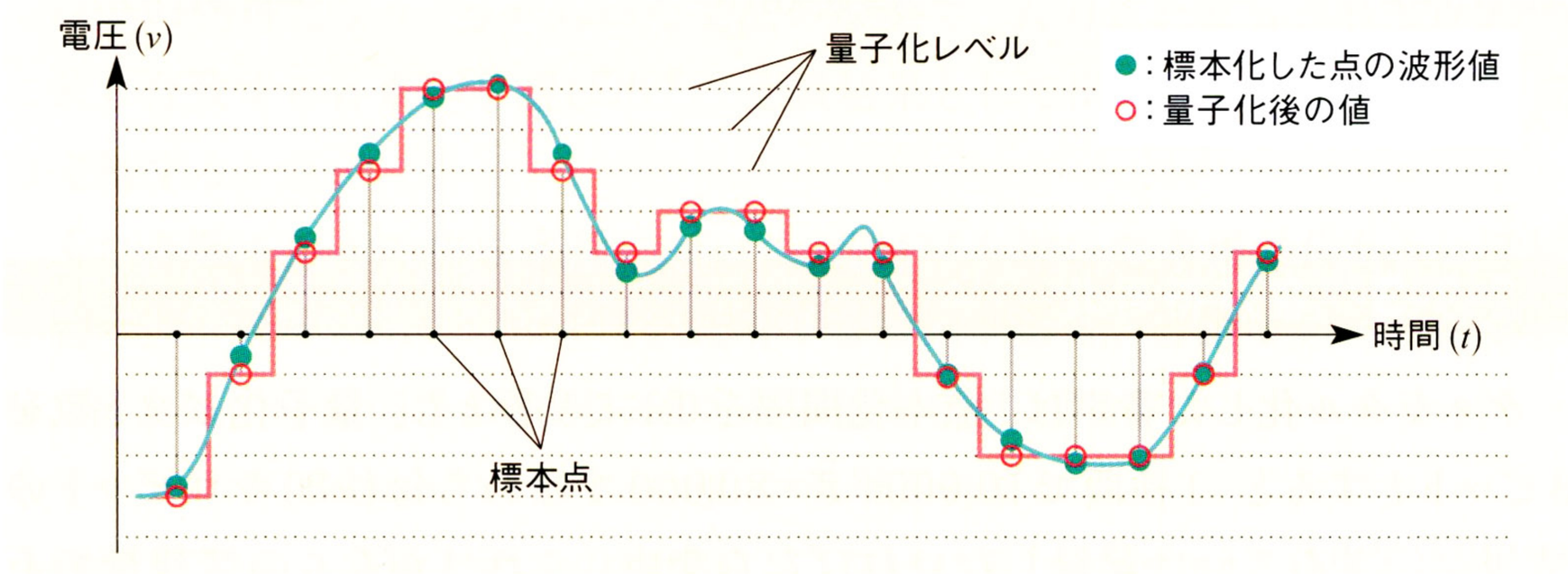
Digital to **A**nalog 変換



Analog信号

担当：高木一幸 takagi@uec.ac.jp

音の標本化と量子化



■図 2.11――信号レベルの量子化

- アナログの音：時間軸と振幅軸に対して連続
- ディジタルの音：時間軸標本化（サンプリング）、振幅軸量子化
- 標本化定理：標本化周波数＝アナログ信号最高周波数 × 2（サンプリング）

音声符号化　データ圧縮

方式	圧縮（ビットレート）	実用例
波形符号化	PCM（Pulse Code Modulation）64kbps	音楽CD
	ADPCM（Adaptive Differential PCM）32kbps	
分析合成符号化	LPC（Linear Predictive Coding）2.4〜4.8kbps	電子楽器
ハイブリッド符号化	CELP（Code Excited Linear Prediction）5.6kbps〜	携帯電話
	EVRC（Enhanced Variable Rate Codec）8.0kbps	
	AMR（Adaptive Multi-Rate）12.2kbps	

担当：高木一幸 takagi@uec.ac.jp

音声音響データの記録メディア

■表R.13——音声ファイルの記録メディア

記録媒体	符号化	サンプリング周波数	量子化ビット数
CD-DA（Compact Disc Digital Audio）	PCM（Pulse Code Modulation）	44.1kHz	16ビット
MD（MiniDisc）	ATRAC（Adaptive Transform Acoustic Coding）	44.1kHz	16, 24ビット
DAT（Digital Audio Tape）	PCM（Pulse Code Modulation）	32kHz, 44.1kHz, 48kHz, 96kHz	16, 24ビット
DVD Audio	MLP（Meridian Lossless Packing）	～192kHz（2チャンネル）, ～96kHz（マルチチャンネル）	16, 20, 24ビット
Super Audio CD（SACD）	Direct Stream Digital（DSD）	2,822.4kHz	—

担当：高木一幸 takagi@uec.ac.jp

主な音声音響ファイルの形式

■表R.14——おもな音声のファイル形式（規格）

形式	説明
MP3 （mp3）	MPEG Audio Layer3。サウンドデータの圧縮フォーマットで，44.1kHz，16ビットステレオのCD品質のサウンドであれば約10分の1にまで圧縮できる。
AAC （m4a, aac）	MPEG-2やMPEG-4で使用される音声圧縮フォーマットで，圧縮率はMP3の約1.4倍で，音質は同程度である。
WMA （wma）	マイクロソフトが開発した音声圧縮方式で，MP3などと同程度の音質で，圧縮率を約1/22まで上げることができる。
AIFF （aif, auなど）	Audio Interchange File Format。オーディオデータの交換に利用されるフォーマットである。
WAVE （wav）	Windowsで使用される標準的なサウンドデータ。音声波形をPCMによってコード化している。22.05kHz，44.1kHzなどのサンプリング周波数，8ビット，16ビットなどの量子化ビット数の設定によって音質が変化する。
MIDI （mid, rmi, smfなど）	音の高さ，強さや音色などの演奏情報や操作情報に基づくデータを伝送するための規格。従来は音源などのMIDI機器を必要としたが，ソフトウェアシンセサイザによりコンピュータのみで再生可能になった。

担当：高木一幸 takagi@uec.ac.jp

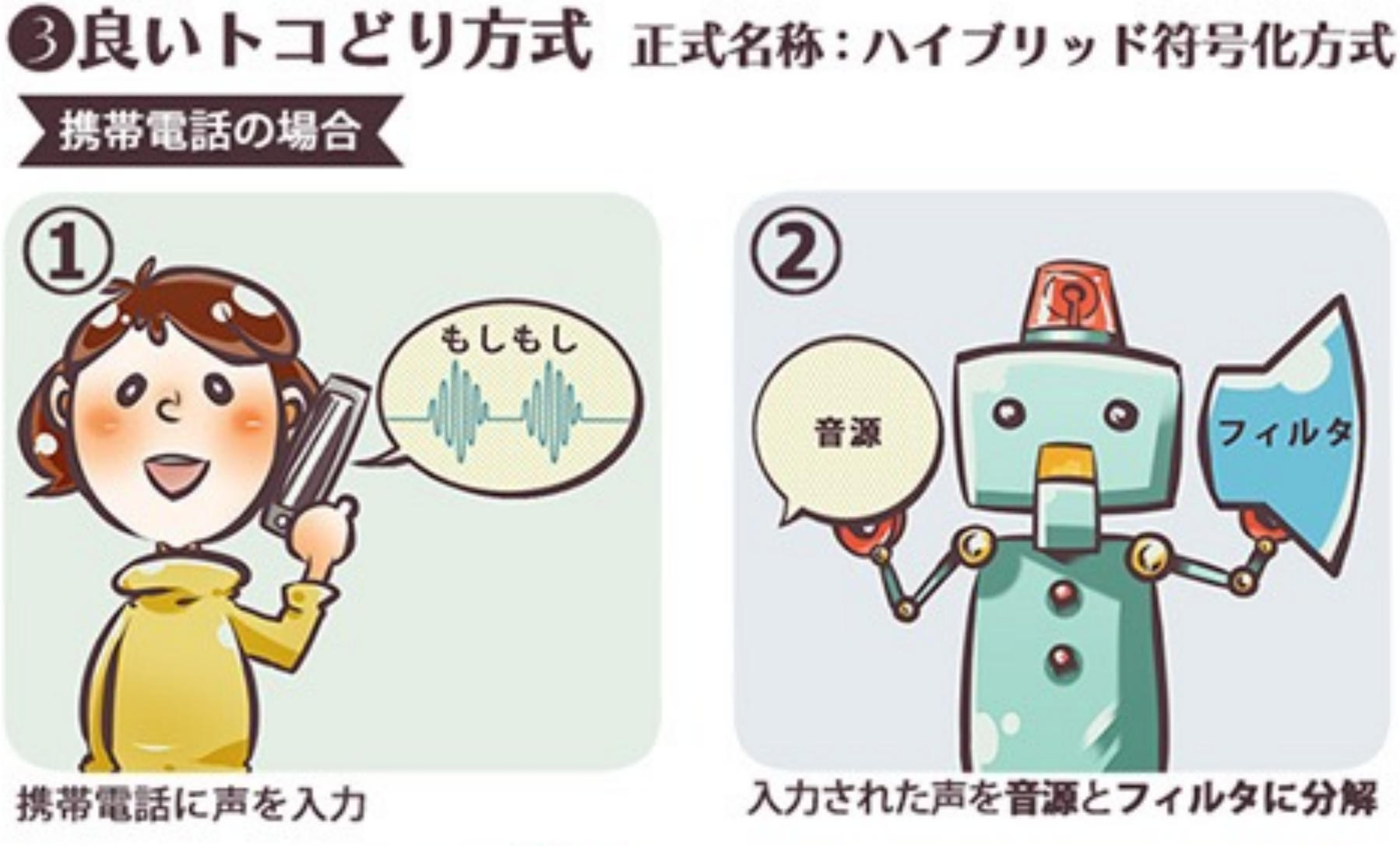
音声符号化　データ圧縮

方式	圧縮（ビットレート）	実用例
波形符号化	PCM（Pulse Code Modulation）64kbps	音楽CD
	ADPCM（Adaptive Differential PCM）32kbps	
分析合成符号化	LPC（Linear Predictive Coding）2.4～4.8kbps	電子楽器
ハイブリッド符号化	CELP（Code Excited Linear Prediction）5.6kbps～	携帯電話
	EVRC（Enhanced Variable Rate Codec）8.0kbps	
	AMR（Adaptive Multi-Rate）12.2kbps	

担当：高木一幸 takagi@uec.ac.jp

出典：徹底究明！「携帯電話の声は、本人の声ではない」説は本当なのか!?

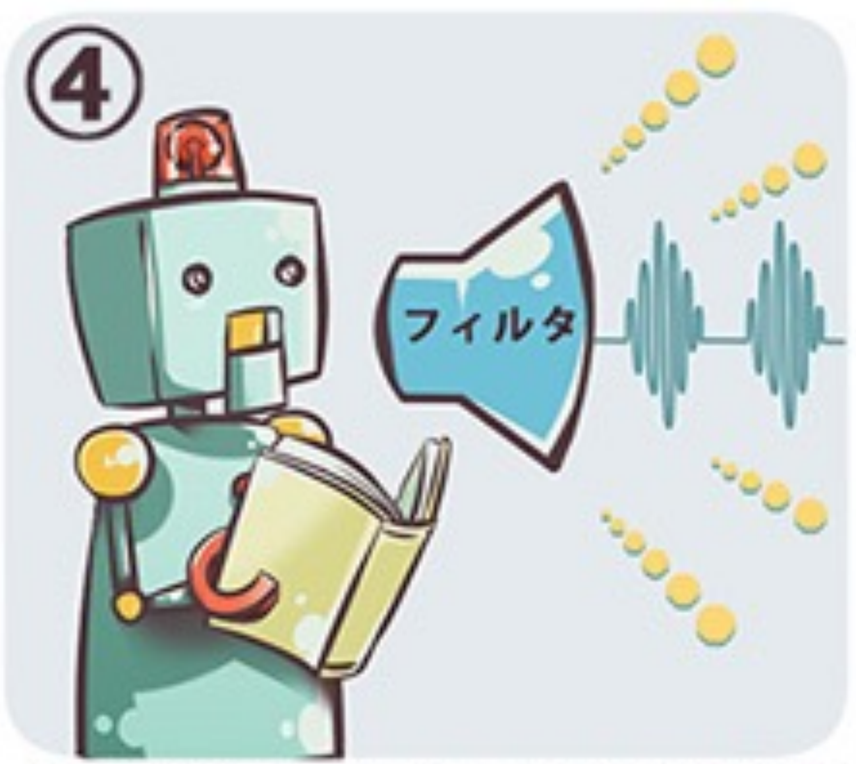
https://time-space.kddi.com/feature/tsushin-chikara-sp/20160404/



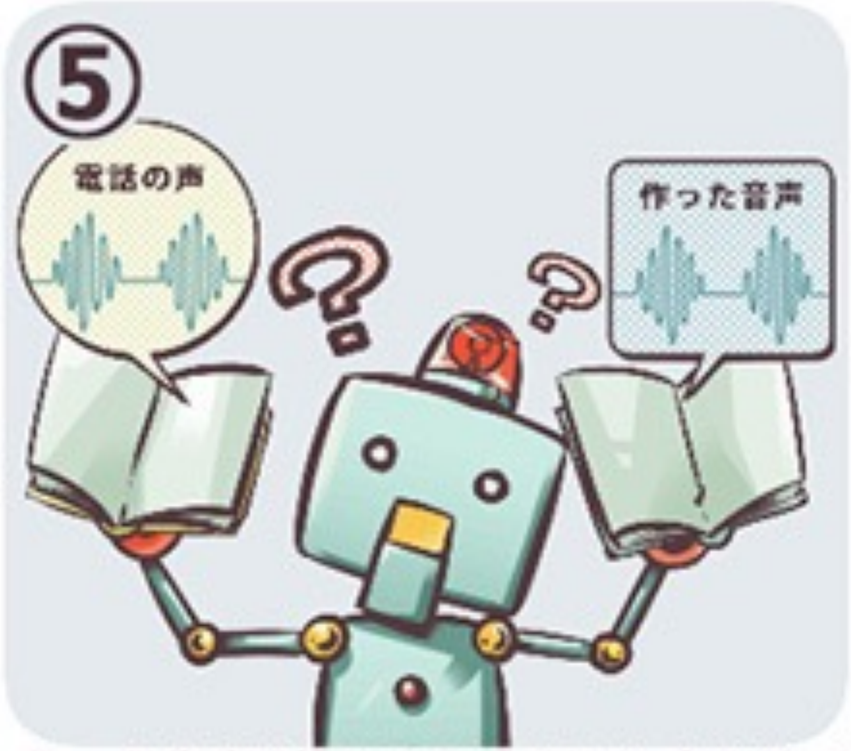
携帯電話に声を入力

入力された声を音源とフィルタに分解

分解した音源を組み立てるパターンを
固定コードブック(音の辞書)から探す



選んだパターンをフィルタに通して
音声(波形)を作る



作った音声(波形)と入力された声を比べ
同じように聴こえるパターンを一瞬ま
えに作られた音声(波形)も使って、
固定コードブック(音の辞書)から選ぶ

携帯電話ではほぼリアルタイムで0.02秒ごとに分析して送っている

作った音声(波形)と入力された声を比べ
同じように聴こえるパターンを一瞬ま
えに作られた音声(波形)も使って、
固定コードブック(音の辞書)から選ぶ

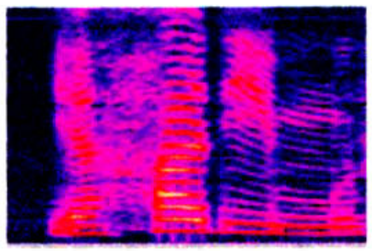
携帯電話ではほぼリアルタイムで0.02秒
ごとに分析して送っている

このような工程を経て、携帯電話の音声
は作られている

ハイブリッド符号化方式

「固定コードブック」には、"音の素"となる組み合わせのパターンが
２の32乗、つまり約43億にもなります。

音声認識システムの全体像



音響モデル

/a/ モデル

/i/ モデル

⋮

/sil/ モデル

音素並び仮説

a, s, u

a, s, u, sh, i, ch, i, j, i

a, s, u, sh, i, ch, i, j, i, sh, u, q, p, a, ts, u

a, s, u, sh, u, q, p, a, ts, u

sh, u, q, p, a, ts, u, d, o, r, u

音響スコア

−1500

−800

−1200

−2500

−2500

単語並び仮説

明日 七時

明日 七時

明日 七時 出発

明日 出発

出発 ドル

言語スコア

−200

−600

−800

−950

−990

スコアの統合

最上位仮説の出力

©西川仁、佐藤智和、市川治著、清水昌平編、テキスト・画像・音声データ分析、2020年5月21日、講談社、ISBN978-4-06-518804-0

図 1.11　音声認識器の全体像

担当：高木一幸 takagi@uec.ac.jp

音素ごとの尤度を出すモデル：音響モデル

音素：発音記号のようなもの

（例） a i u k s ch … sil（実際にはもっと詳細な音素定義）

尤度：想定する音素に対しての入力の尤もらしさ

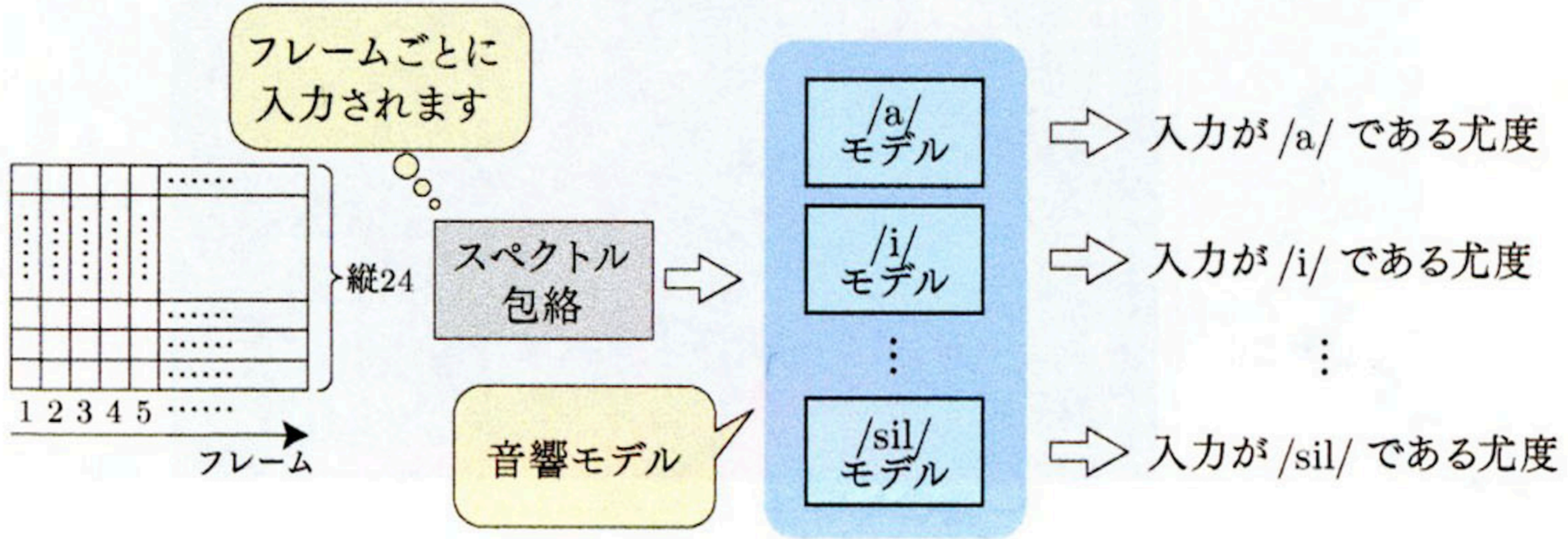


図 1.4 音響モデル

西川仁、佐藤智和、市川治著、清水昌平編、テキスト・画像・音声データ分析、2020年5月21日、講談社、ISBN978-4-06-518804-0

音素ごとの尤度を出すモデル：音響モデル

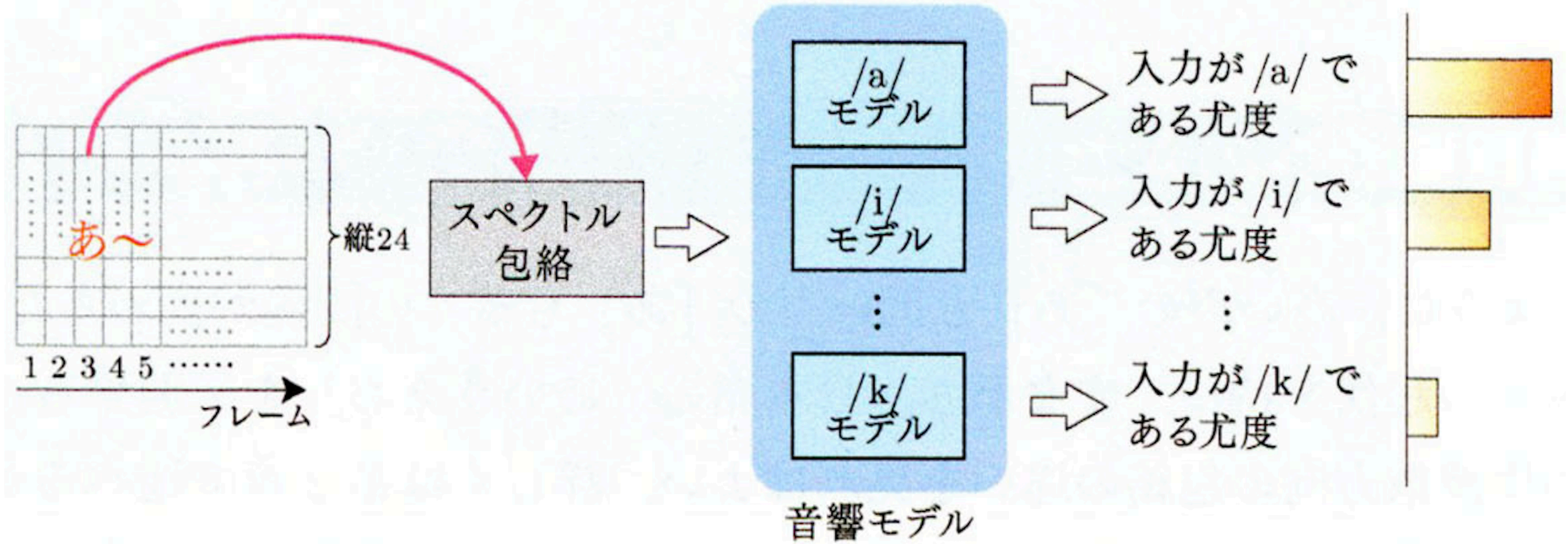


図 1.5　/a/ にあたる部分を発声

文全体の音響尤度

「あき」と発声した時に，それが音素列 /a/ /k/ /i/ である尤度は，
全フレームの尤度を集めたもの

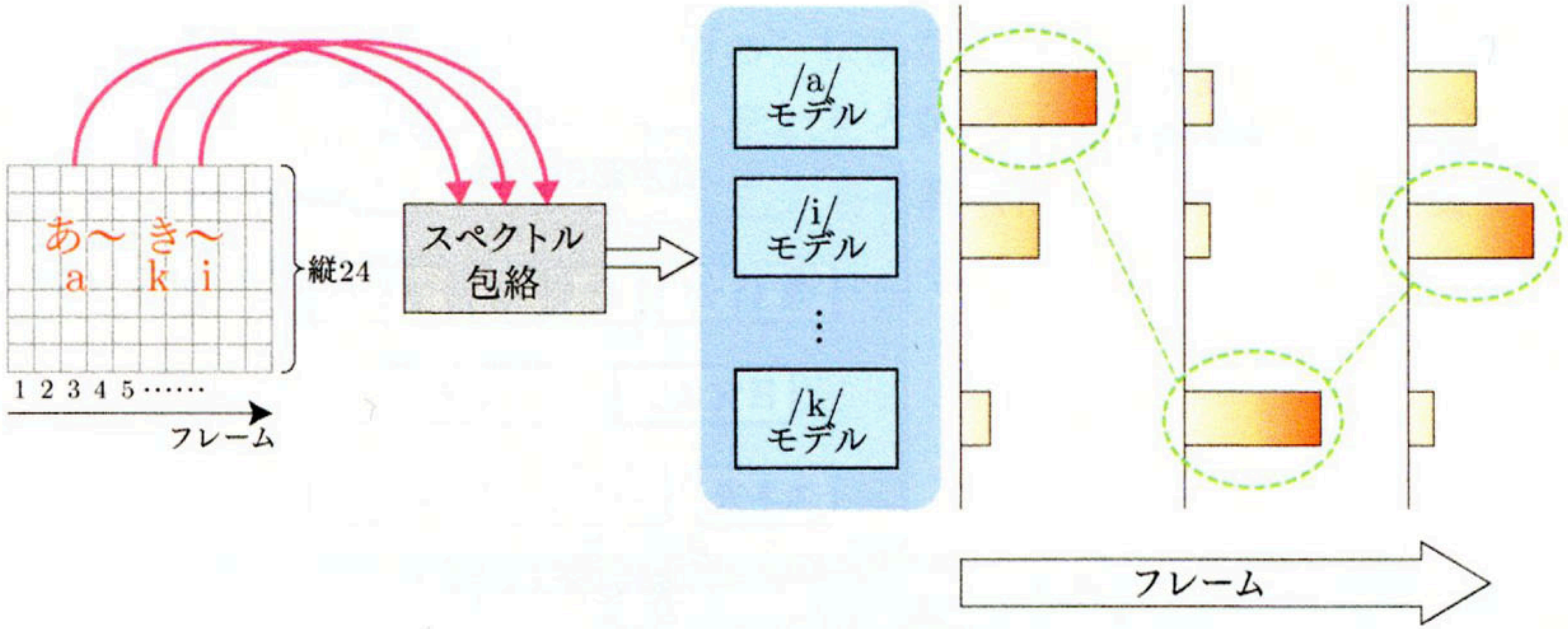


図 1.6 音素列の仮説 /a/ /k/ /i/ に対する尤度

担当：高木一幸 takagi@uec.ac.jp

文全体の音響尤度

「あき」と発声した時に，それが間違った音素列 /k/ /a/ /i/ である尤度は，
正しい音素列だと思って計算した尤度よりも小さくなる。

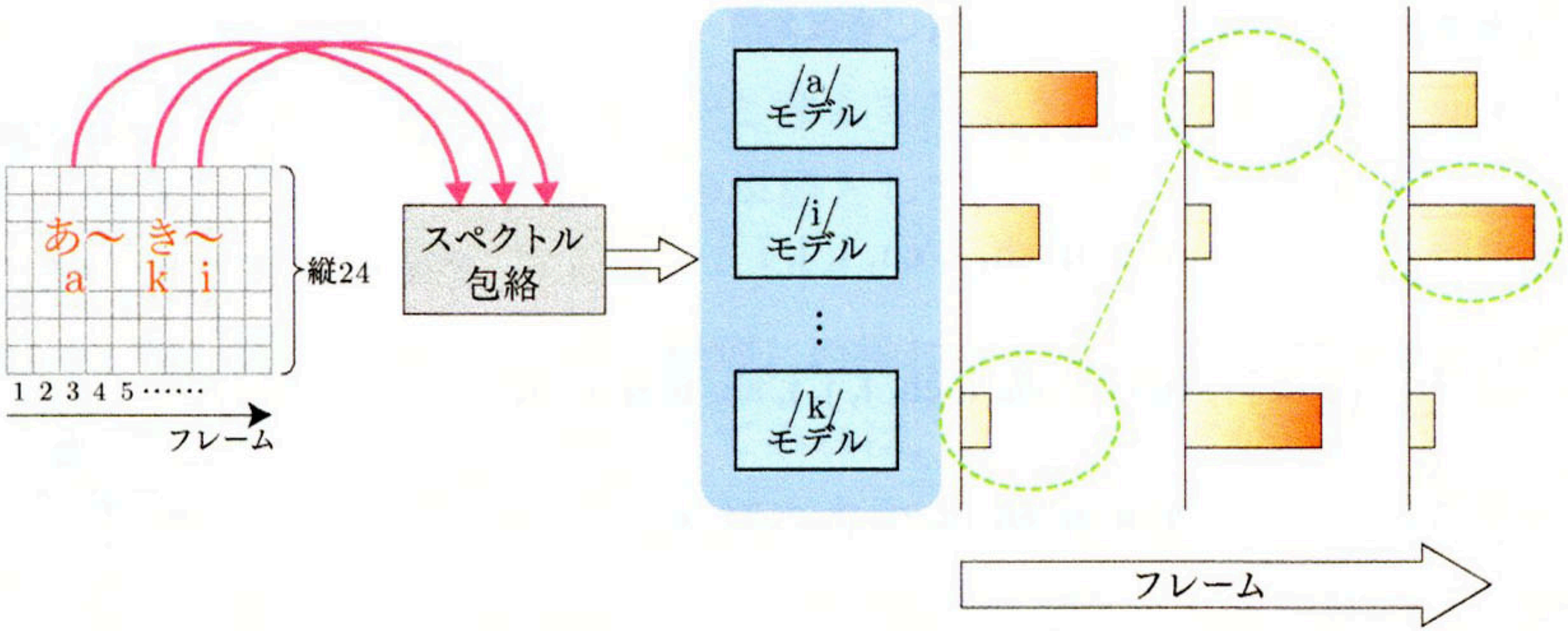
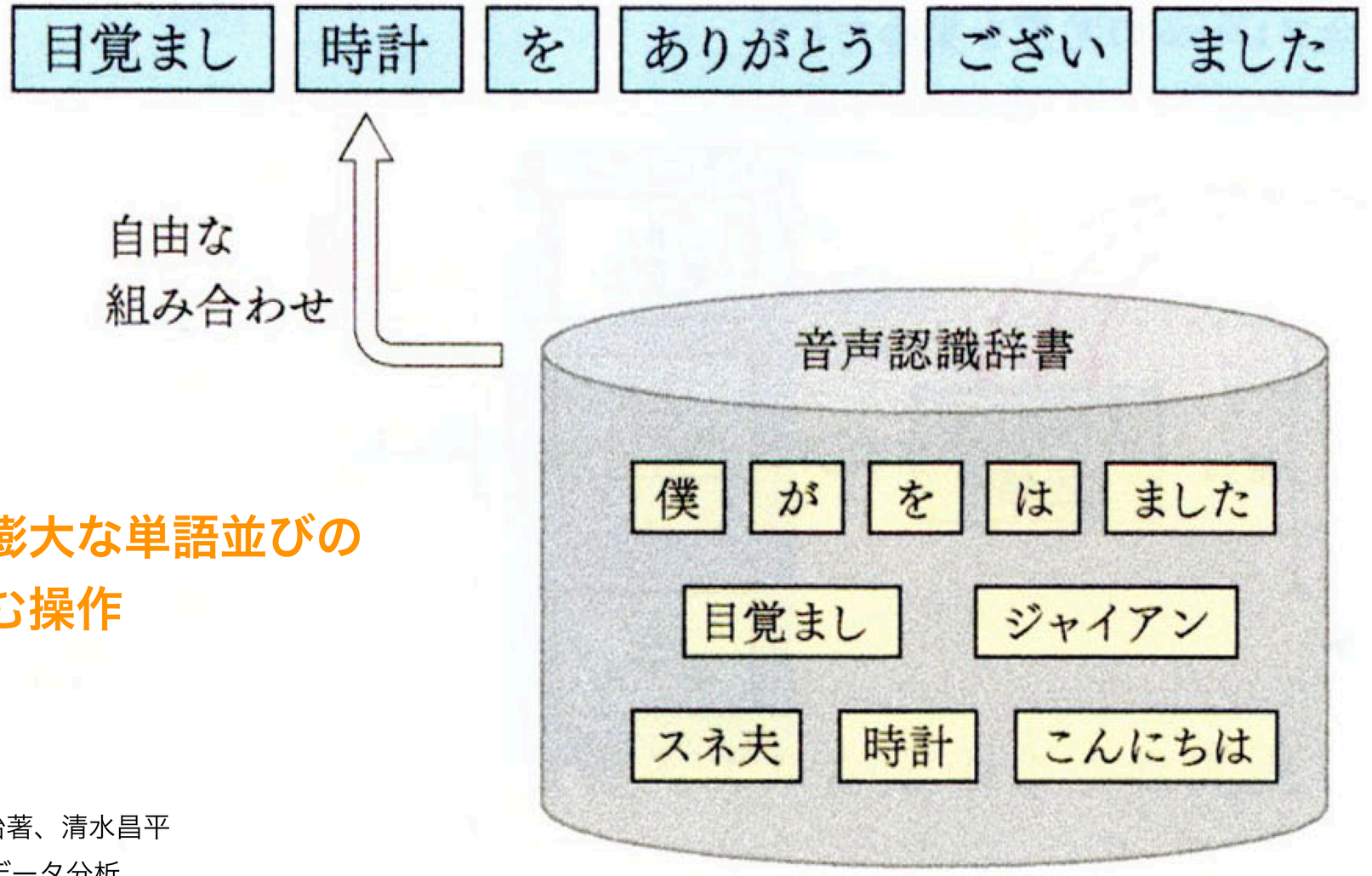


図 1.7 音素列の仮説 /k/ /a/ /i/ に対する尤度

音声認識における単語並びの仮説



音声認識とは膨大な単語並びの候補を絞り込む操作

図 1.8　音声認識辞書

担当：高木一幸 takagi@uec.ac.jp

音声認識における単語並びの仮説

例えば、「ドル」「出発」「七時」「明日」の４単語が辞書にあったとする。
可能な単語並びは？

（例） 単語並び	音素並び
明日 七時	a, s, u
明日 七時	a, s, u, sh, i, ch, i, j, i
明日 七時 出発	a, s, u, sh, i, ch, i, j, i, sh, u, q, p, a, ts, u
明日 出発	a, s, u, sh, u, q, p, a, ts, u
出発 ドル	sh, u, q, p, a, ts, u, d, o, r, u

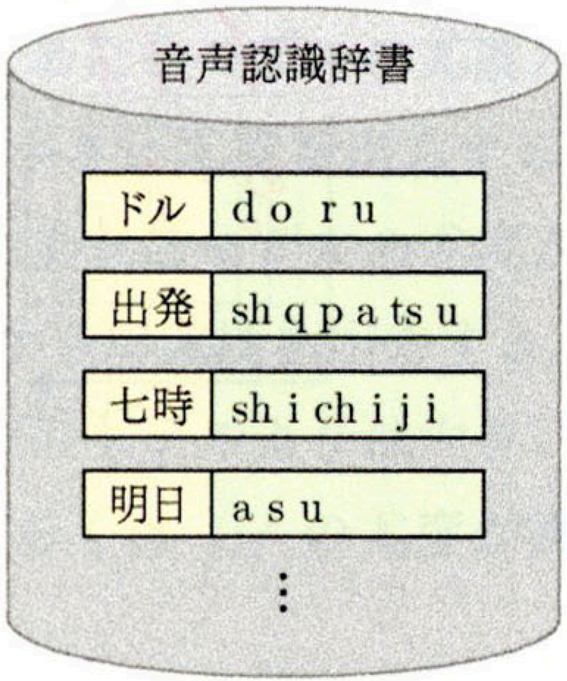


図 1.9　単語並びと音素並び

音声認識のための言語モデル

先行2単語

（例） 内閣 総理 この後にどの単語が来そうか？

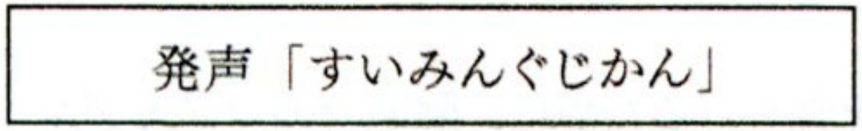
大事 大臣

- 「大臣」も「大事」もありうるが，
  「大臣」の方が「大事」よりも確率が高い。

- 膨大な数の単語並びの仮説があったとしても，
  出現する確率が高い並びと，低い並びがある。

図 1.10 先行単語の影響

音響モデルと言語モデルの綱引き



音響モデル	言語モデル
スイミング と言ったはず！	「スイミング時間」 なんて言葉はありえない！
「スイミング時間」でしょ	「睡眠時間」でしょ

明瞭に発声した時　音響モデル　優勢 ⇨ 結果出力「スイミング時間」

不明瞭に発声した時　言語モデル　優勢 ⇨ 結果出力「睡眠時間」

図 1.12　音響モデルと言語モデルのバランス

テキスト・画像・音声データ分析、2020年5月21日、講談社、ISBN978-4-06-518804-0

スペクトルデータ：波形データの変換

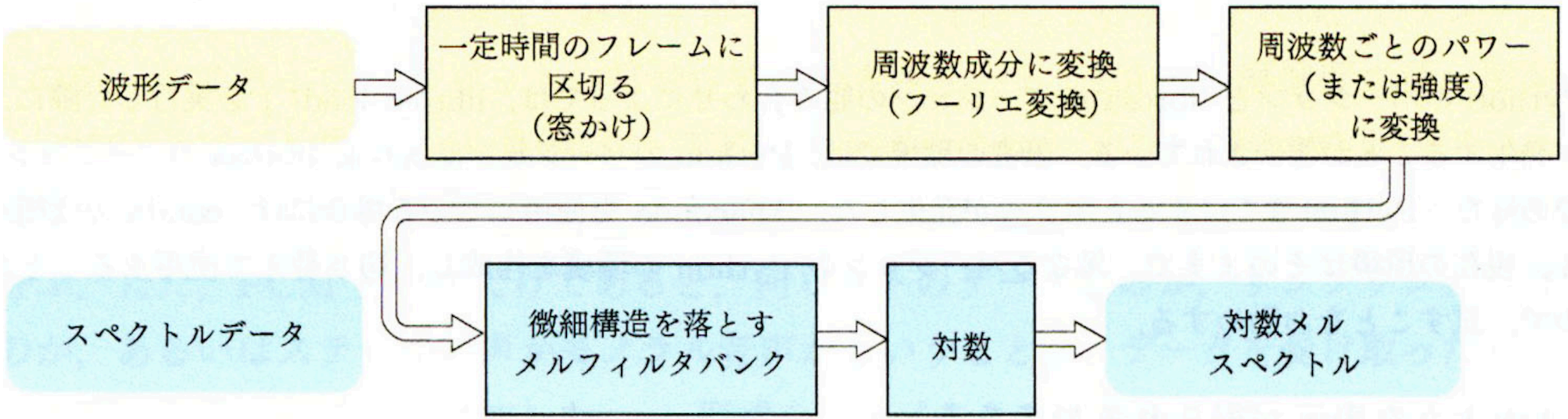


図 2.3　波形データの変換

担当：高木一幸 takagi@uec.ac.jp

スペクトルデータ：窓掛け操作

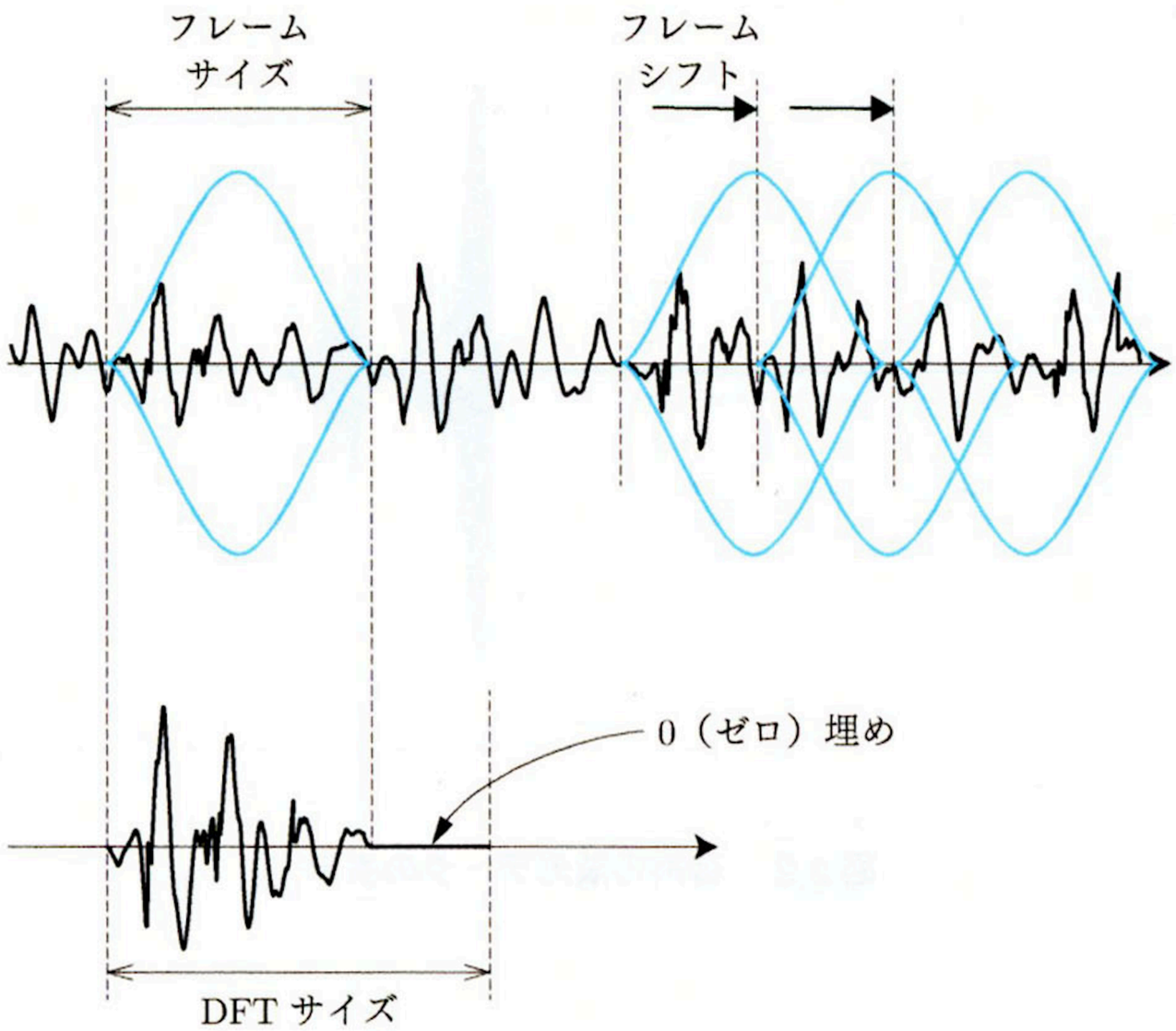


図 2.4　窓掛け操作

担当：高木一幸 takagi@uec.ac.jp

音響モデル：単音素

a	母音
i	
u	
e	
o	
a:	長母音
i:	
u:	
e:	
o:	

w
y
j
my
ky
dy
by
gy
ny
hy

ry
py
p
t
k
ts
ch
b
d
g

z
m
n
s
sh
h
f
r

N	撥音 「ん」
q	促音 「っ」
sp	単語間の 短い無音
silB	文頭の 無音
silE	文末の 無音

図 3.1　単音素（モノフォン）定義の例

音素に対応する音声は前後の音素に大きな影響を受ける

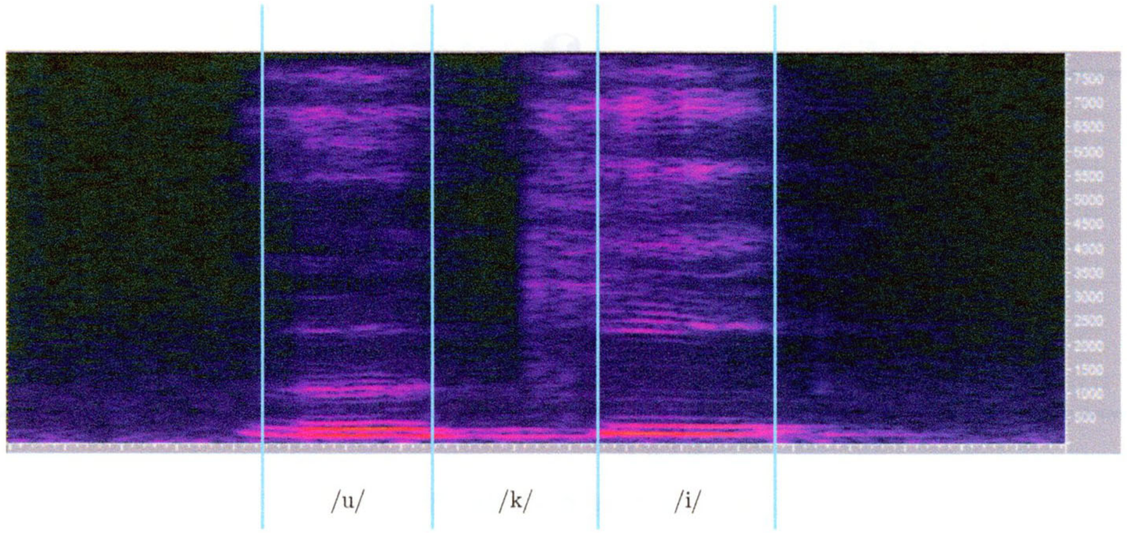


図 3.2　「うき」（/u/ /k/ /i/）の発声のスペクトログラムの例

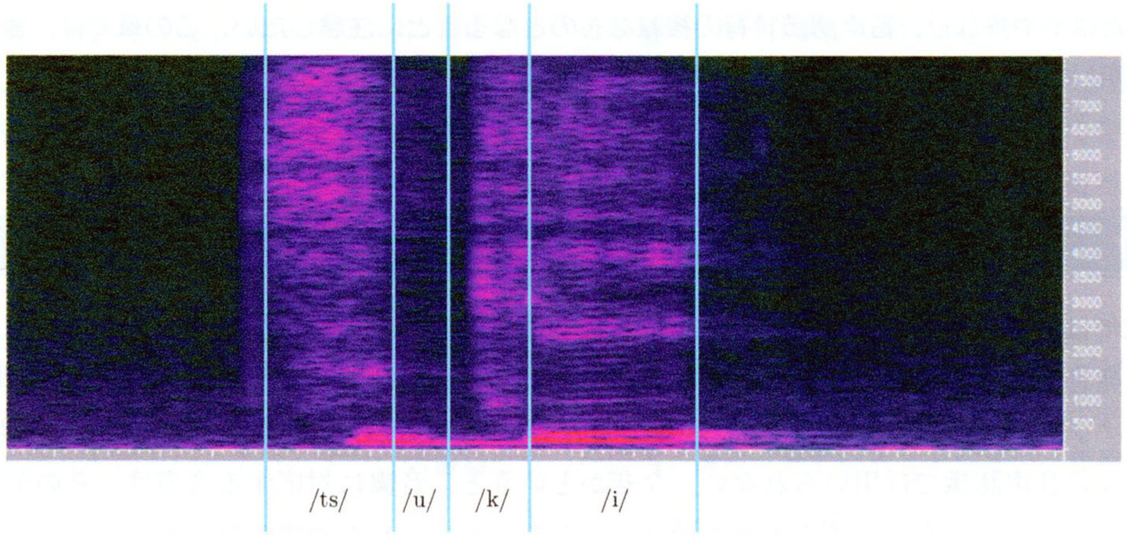


図 3.3　「つき」（/ts/ /u/ /k/ /i/）の発声のスペクトログラムの例

/u/

- うき /uki/ の発音では /u/ のスペクトルが明瞭
- つき /tsuki/ の発音では /u/ のスペクトルが消失（無声化）

音響モデル精度向上のため
自分自身を含む前後3音素を
考慮…トライフォン triphone

トライフォン（triphone）

単語列	「つき」
モノフォン列	/ts/ /u/ /k/ /i/
トライフォン列	/ ts+u / / ts-u+k / / u-k+i / / k-i /

図 3.5　コンテキスト依存音素（ここではトライフォン）の例

単語列	「つき」 「が」 「でた」
モノフォン列	/sil/ /ts/ /u/ /k/ /i/ /g/ /a/ /sp/ /d/ /e/ /t/ /a/ /sil/
トライフォン列	/sil+ts/ /sil-ts+u/ /ts-u+k/ /u-k+i/ /k-i+g/ /i-g+a/ /g-a+sp/ /a-sp+d/ /sp-d+e/ /d-e+t/ /e-t+a/ /t-a+sil/ /a-sil/

図 3.6　複数単語からなるトライフォンの例

音素クラスタリング

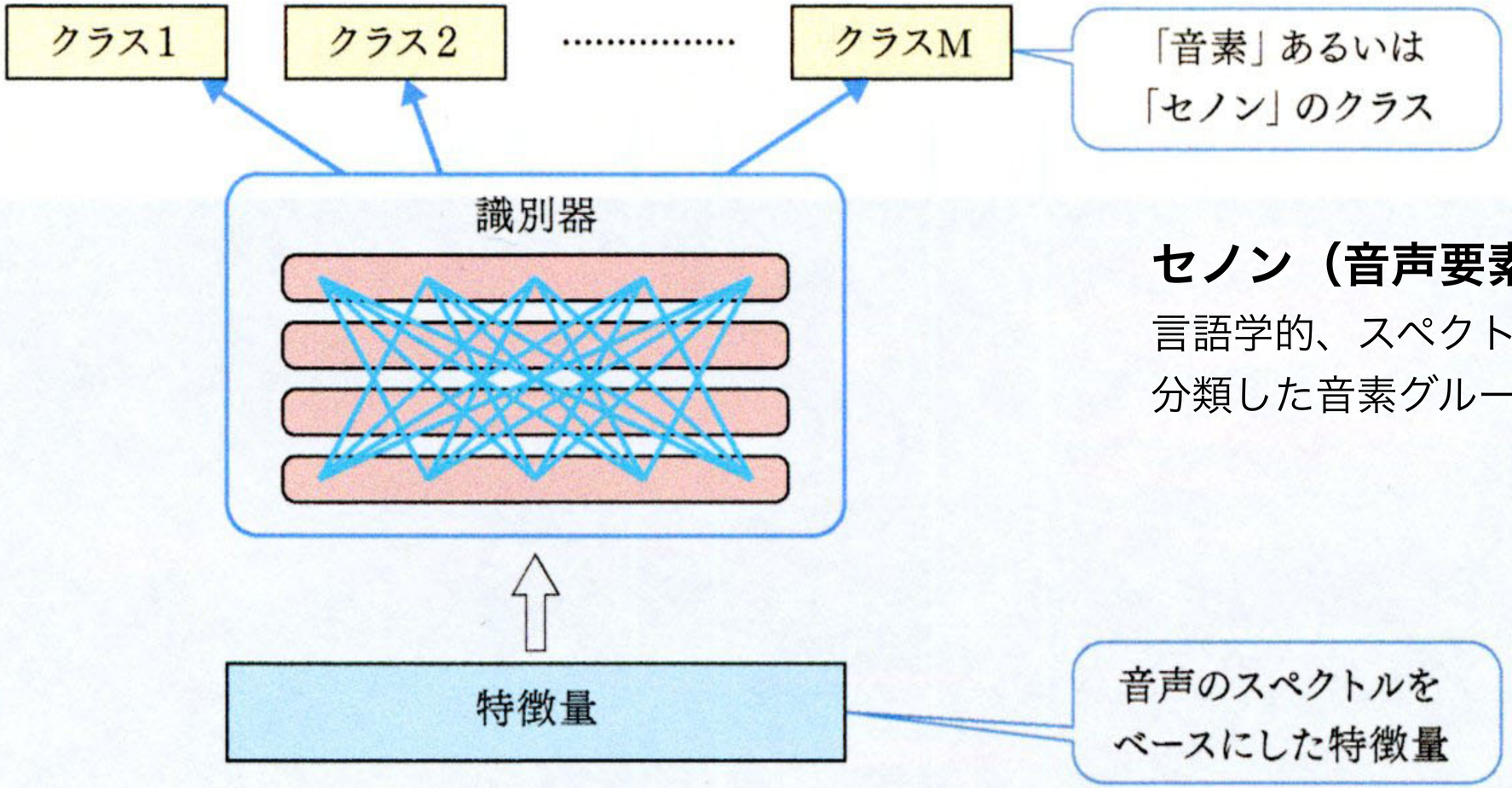


図 3.8　ニューラルネットワーク音響モデルの模式図

セノン（音声要素）

言語学的、スペクトル的な観点で
分類した音素グループ

全結合ニューラルネットワークによる音響モデル

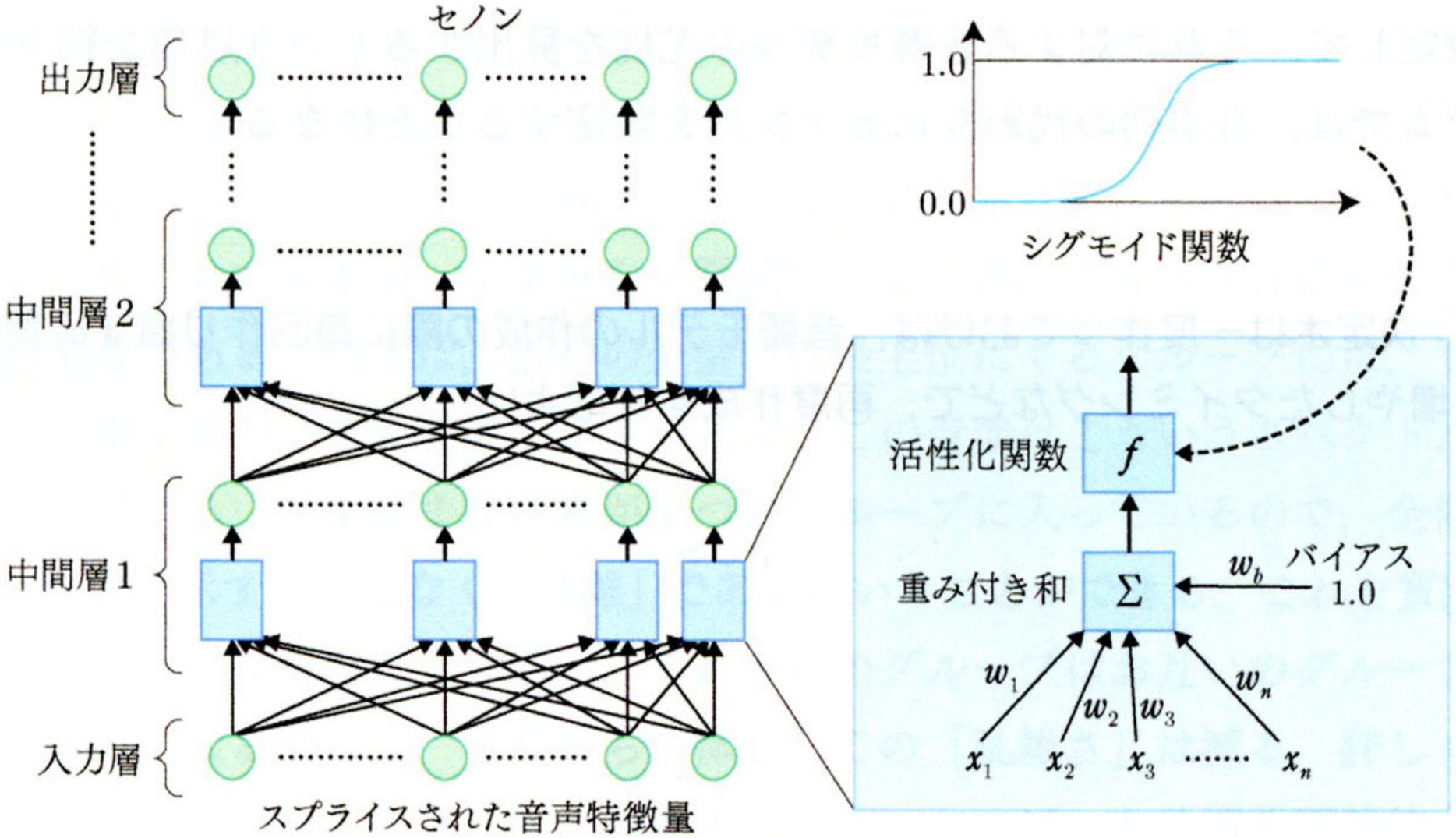
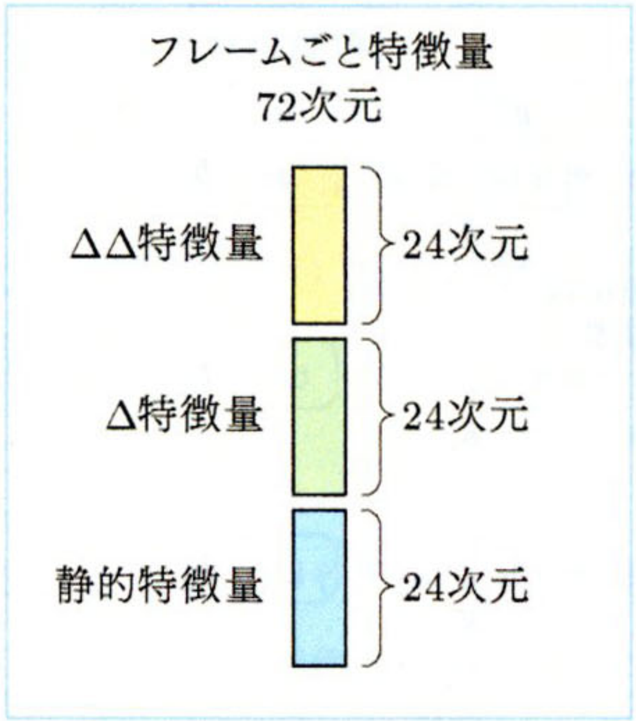
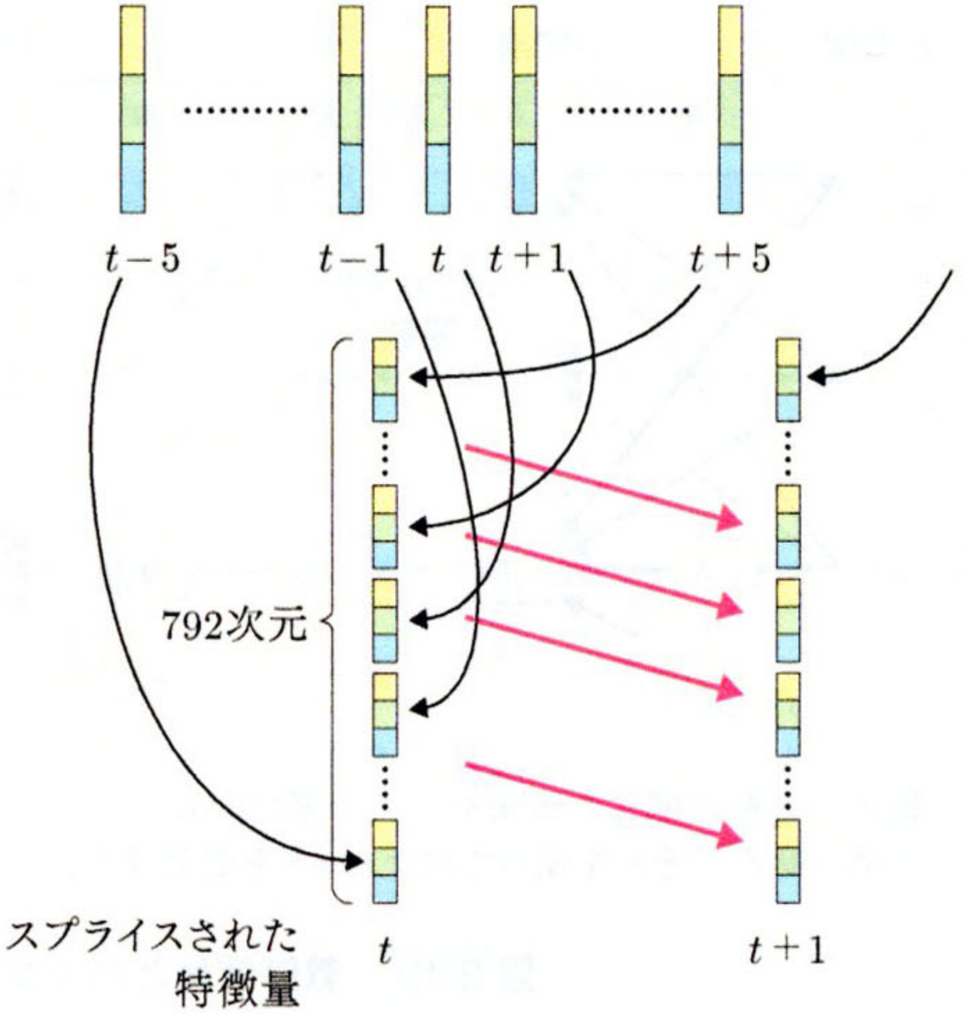


図 3.10　音響モデル（全結合ニューラルネットワーク）

音声特徴量のスプライス（継ぎ合わせ）





前後の複数フレームを
結合した特徴量

図 3.11　スプライスされた特徴量の例

End-to-End音声認識モデル

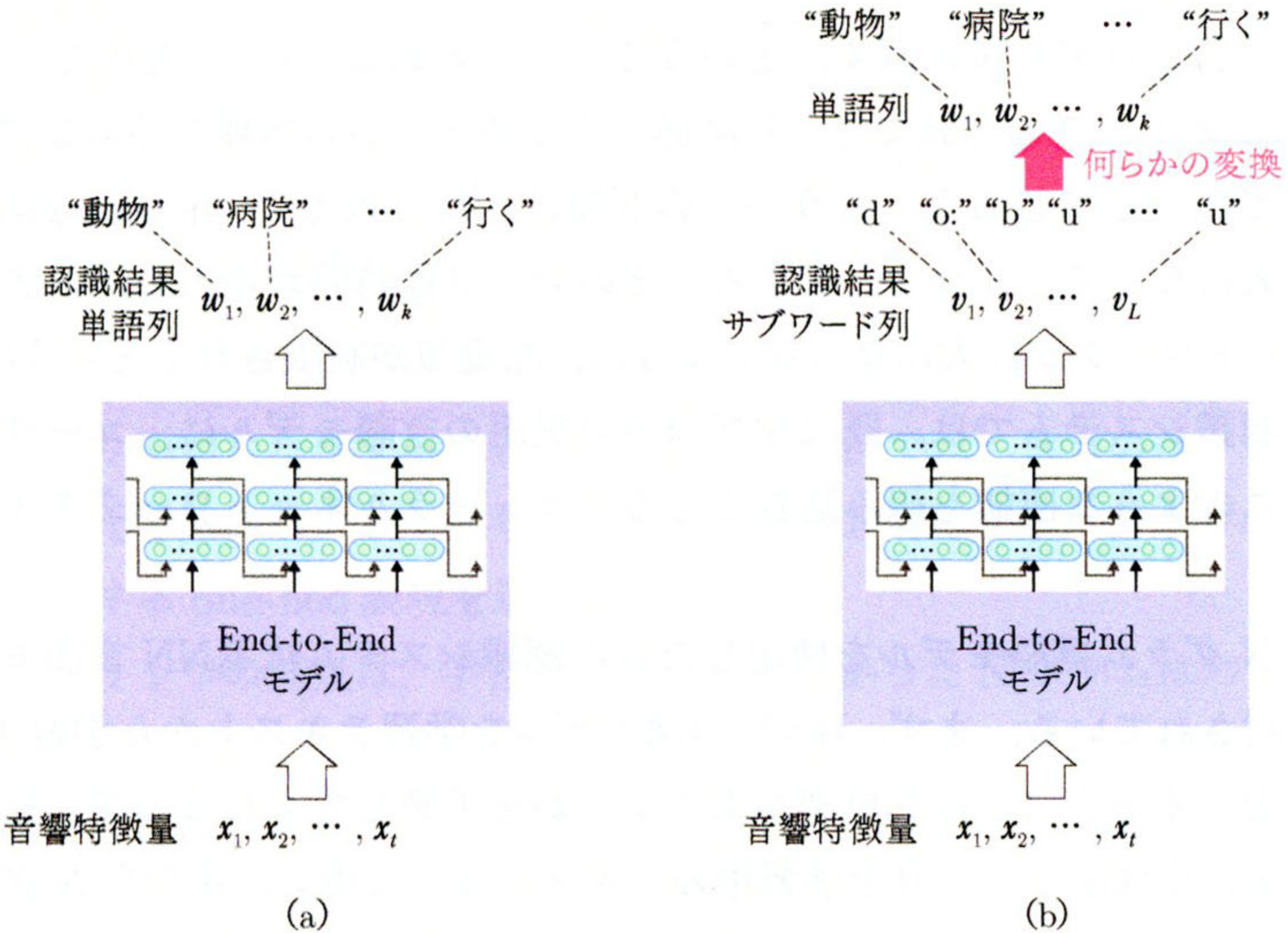


図 6.6 End-to-End 音声認識モデル

担当：高木一幸 takagi@uec.ac.jp

音声合成とは

音声を計算機的に作り出す（合成する）技術の総称

- **分析合成**

  音声を入力とするもの、e.g. ***WORLD***

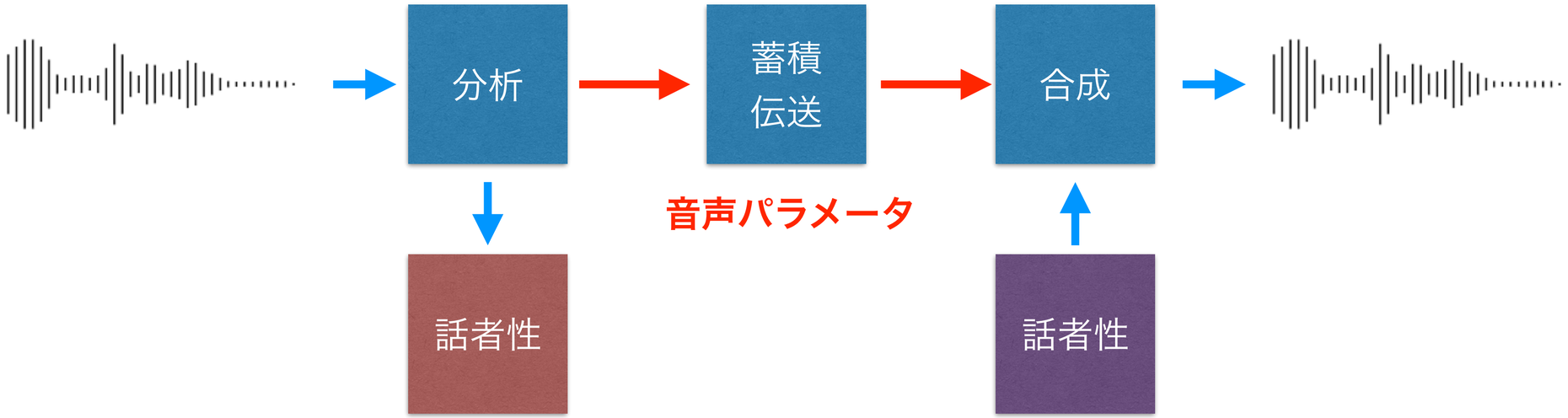
- **テキスト音声合成（Text-To-Speech: TTS）**

  テキストを入力とするもの、e.g. WaveNet

WORLD

WaveNet

音声の分析合成のメリット



- 高能率圧縮伝送：原音声=64kbps、音声パラメータ：3.2kbps
- 音声特徴抽出
- 声質変換
- 代表的なソフトウエア：WORLD

WORLD

録音再生方式・録音編集方式

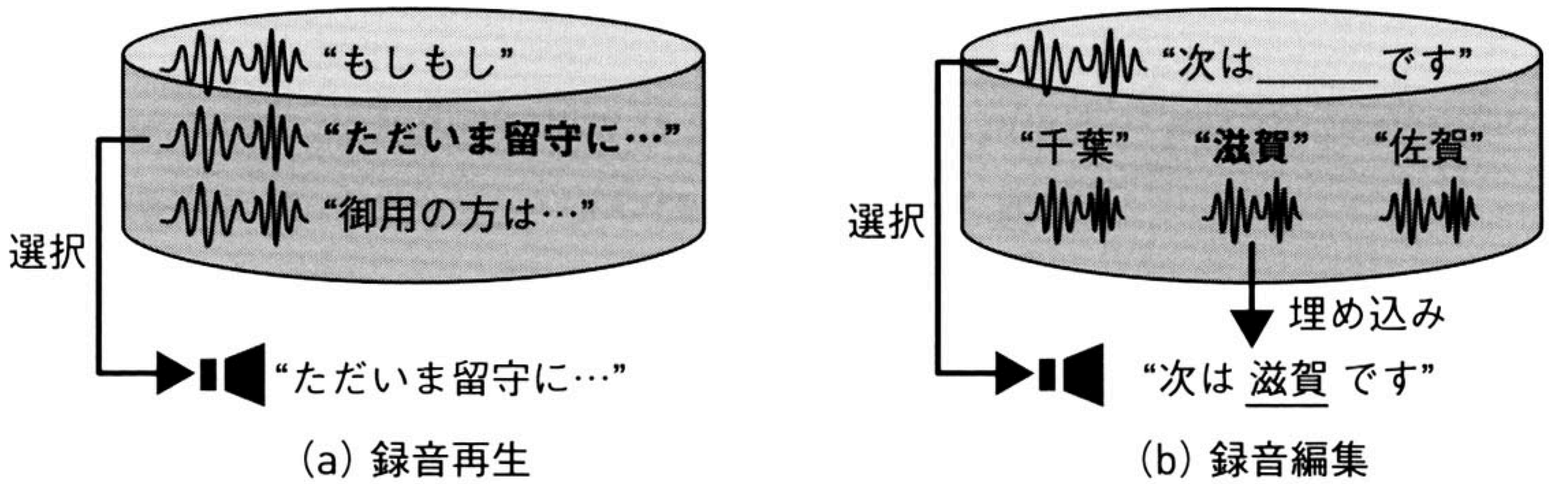
録音編集方式

- 人間が発声した音声を蓄積し、適宜連結して再生
- 電話案内、駅の案内放送
- 単語や文の種類が限られる

規則合成方式

- 音素や音節などの短い単位をコンピュータで合成し連結
- 任意の文字列から音声を生成可能
- 音声認識と組み合わせた対話技術に必須

図 1-2 録音再生方式と録音編集方式

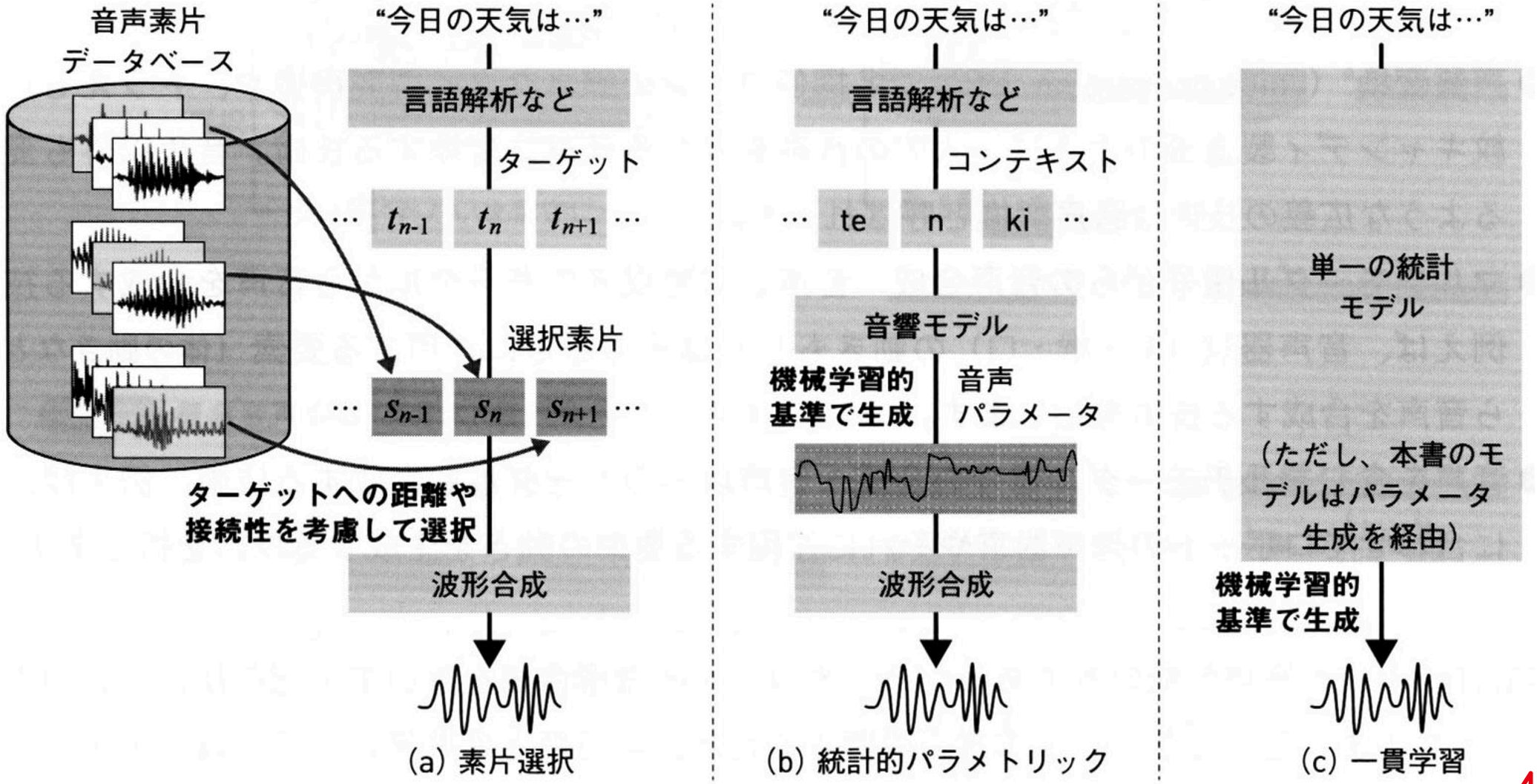


Vocaloid：**素片編集方式**

剣持秀紀氏
解説記事

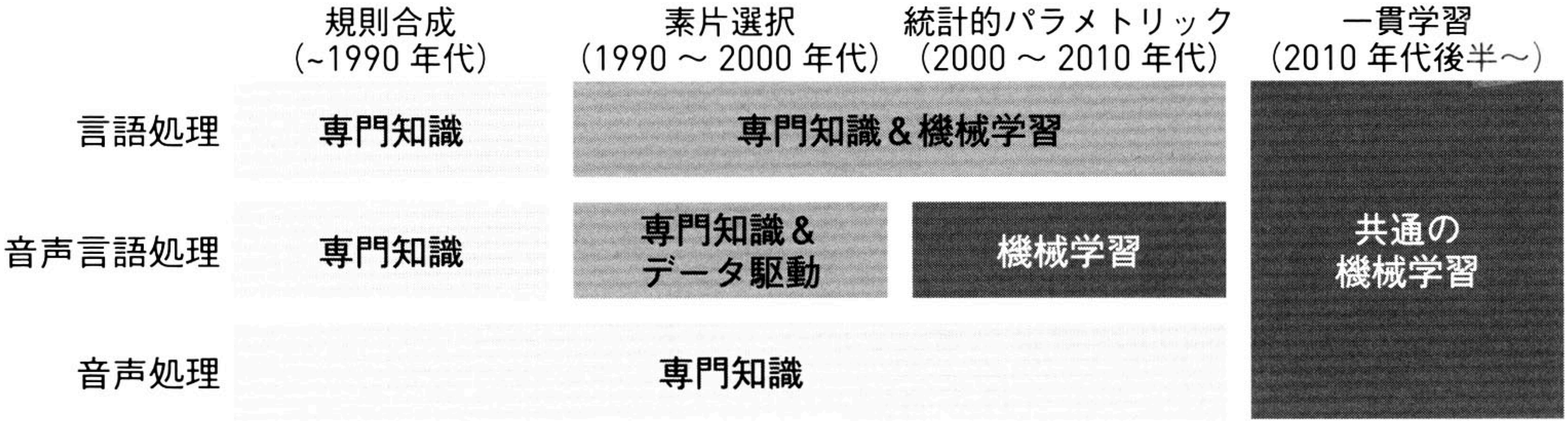
テキスト音声合成方法の比較

図 1-5 素片選択型合成法、統計的パラメトリック音声合成法、一貫学習に基づく合成法の比較



音声合成方式の歴史

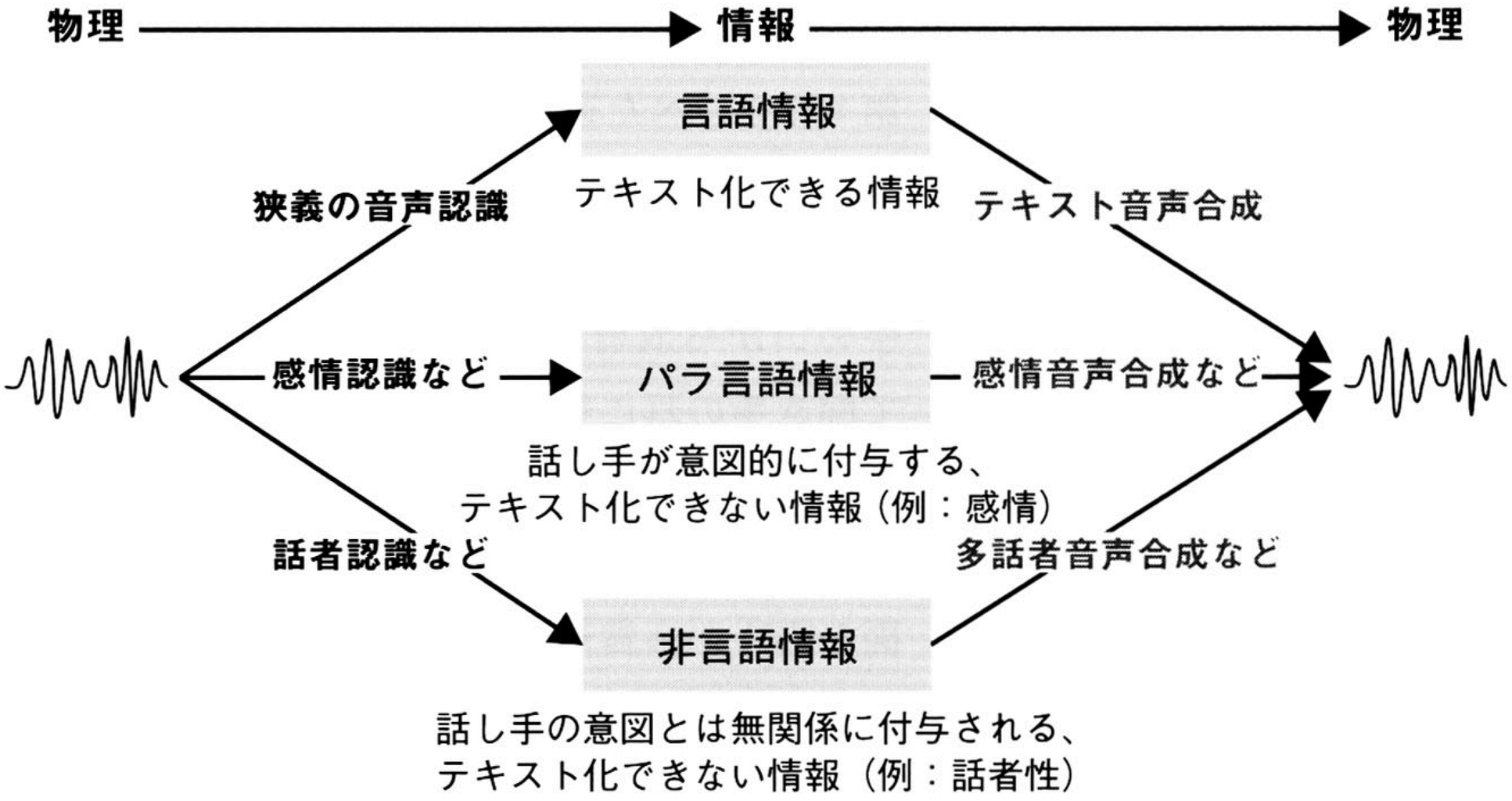
図 1-6　規則合成方式から一貫学習方式における、専門知識からの脱却



©山本龍一、高道慎之介、Pythonで学ぶ音声合成、2021年8月12日、インプレス

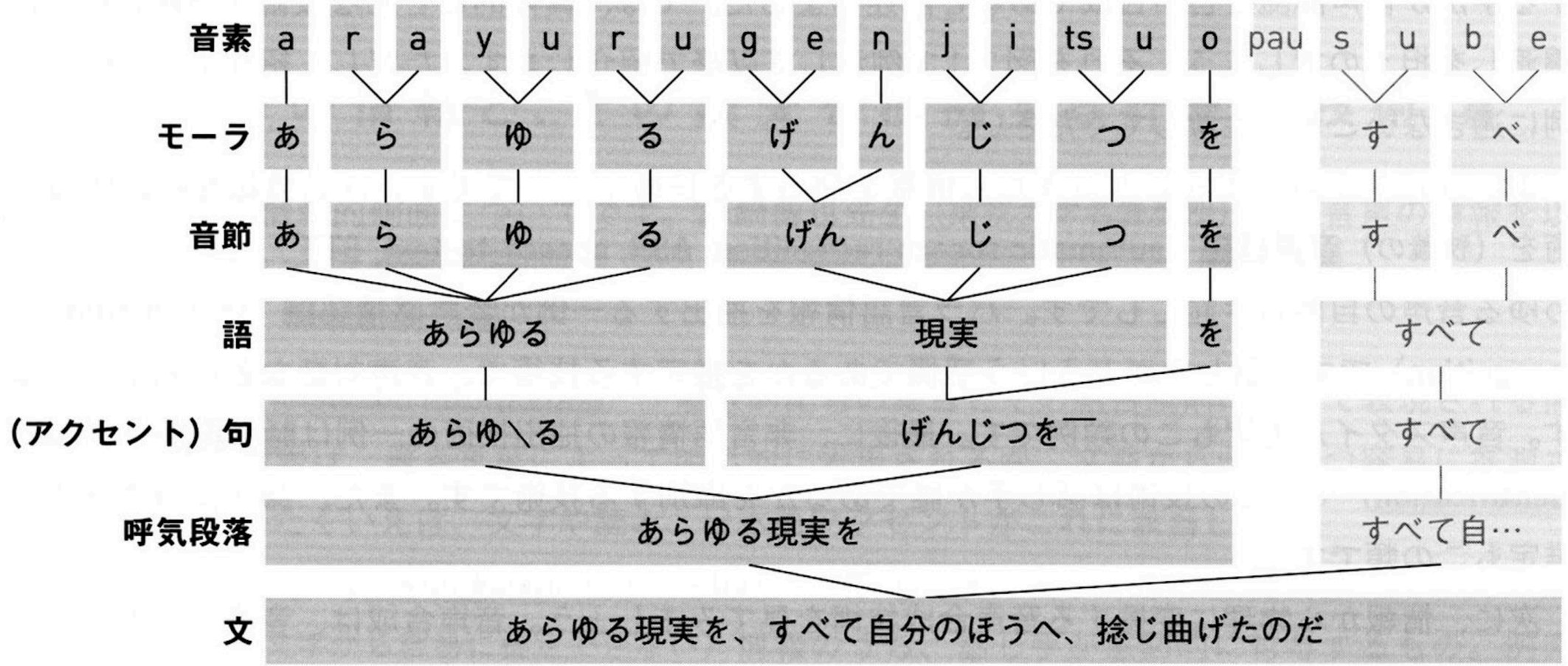
音声の内包する情報

図 2-1 音声の持つ情報と、その授受を計算機で模擬する音声認識合成技術



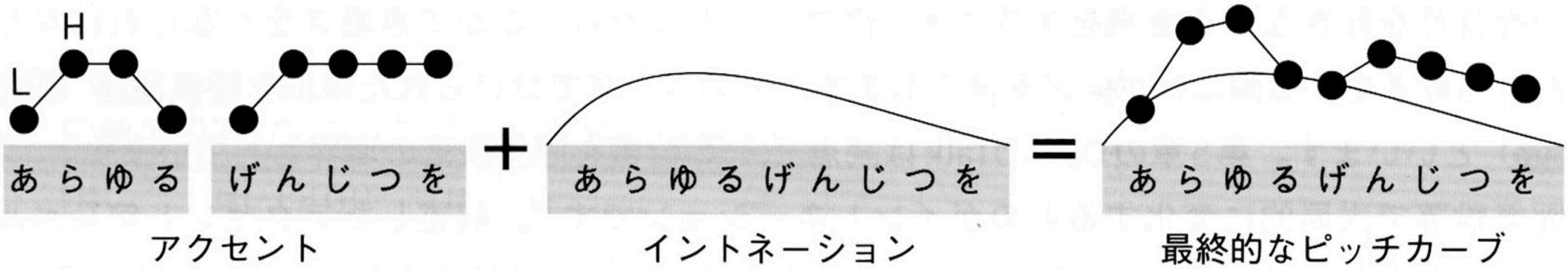
音声言語の階層

図 2-2 言語から推定可能なコンテキストの例。ただし、第5章で扱うコンテキストラベルに、音節や語レベルのものは含みません。この例文は、ATR音素バランス503文[6]という古典的なコーパスに含まれる冒頭文です。



イントネーションとアクセント

図 2-4 アクセントとイントネーションの和で構成されるピッチカーブ。アクセントを音階式で表示しています。



イントネーション

呼気段落（息継ぎの単位）で大局的に変化するピッチカーブ
（主として生理的）

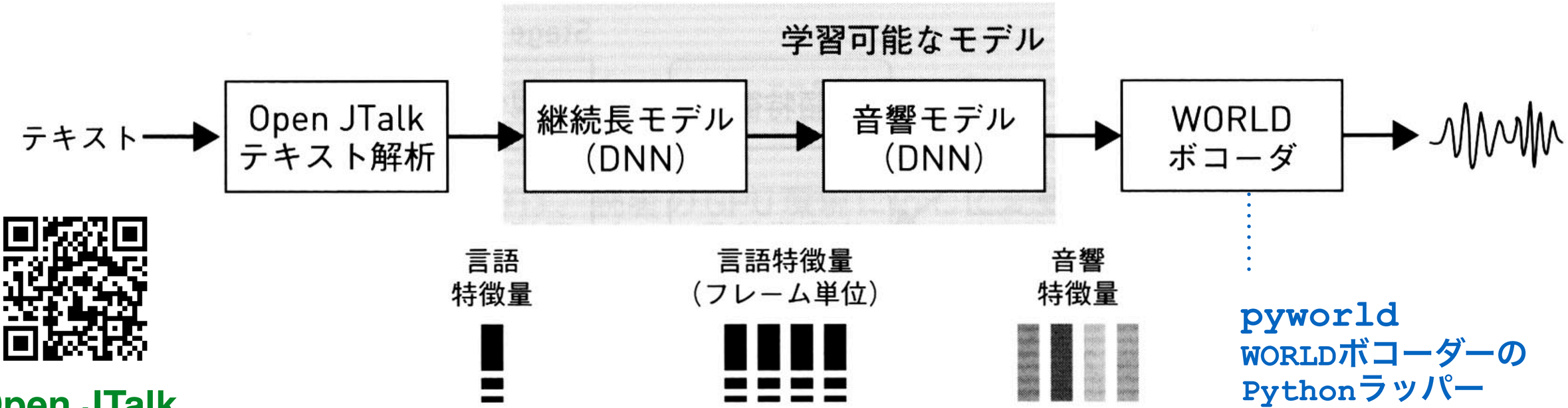
アクセント

単語が持つ高さの動き（単語の意味に関わる）

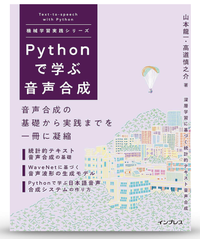
担当：高木一幸 takagi@uec.ac.jp

DNN音声合成の実装

図 6-1 DNN音声合成に基づく日本語音声合成のフロー



Open JTalk

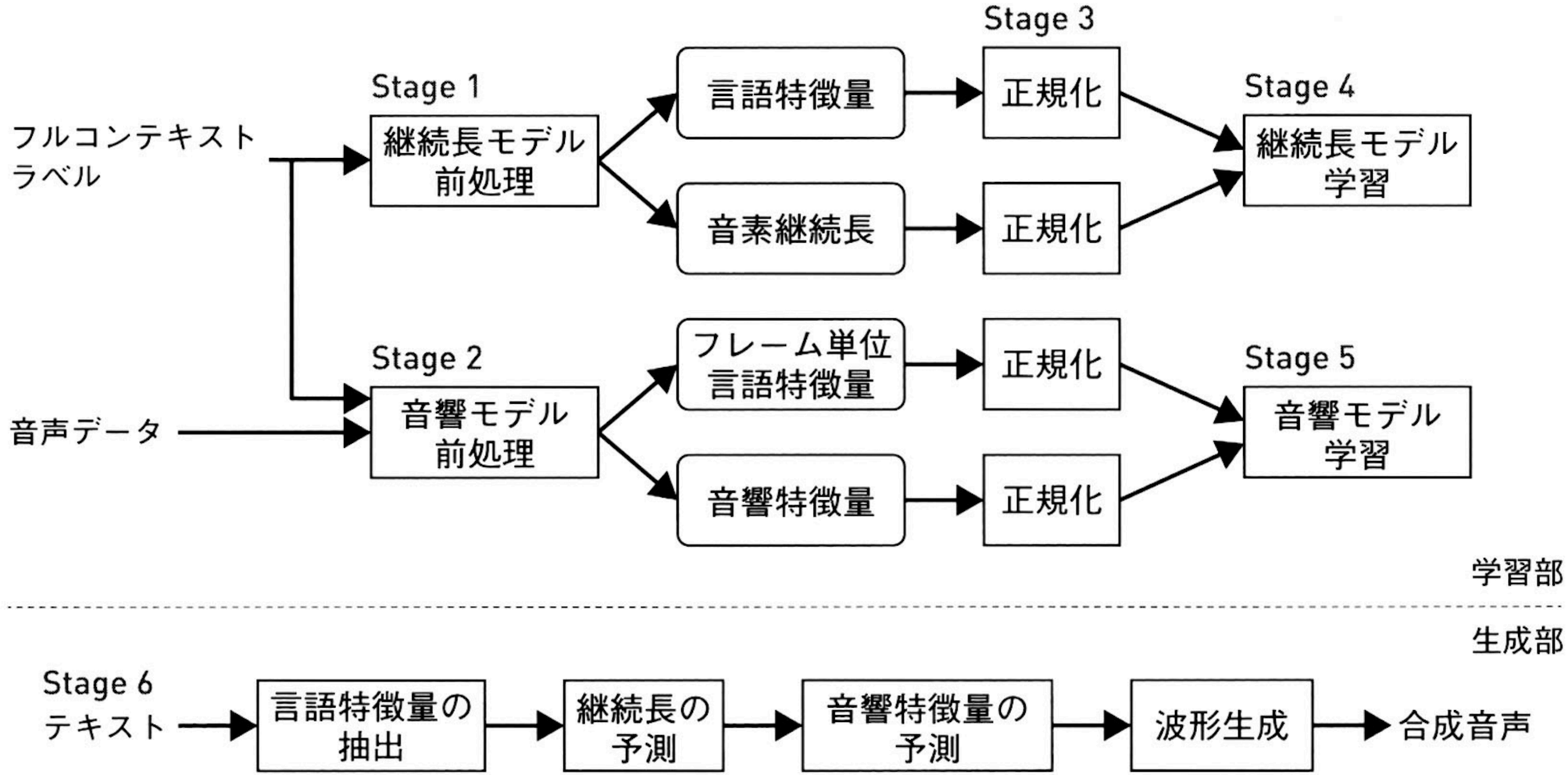


で紹介されているPythonによる日本語音声合成の処理の流れ

©山本龍一、高道慎之介、Pythonで学ぶ音声合成、2021年8月12日、インプレス

DNN音声合成の実装

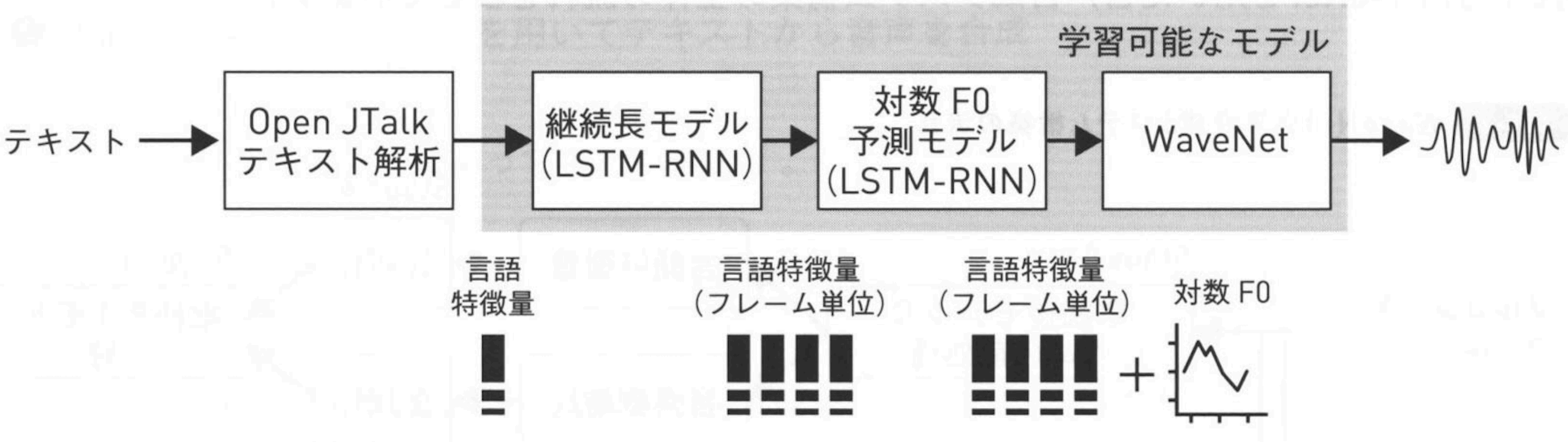
図 6-2 DNN音声合成システム構築の流れ



WaveNet

- 深層学習に基づく音声波形生成モデル（2016年９月）。
- 音声生成過程を仮定せず、音声を１次元の数値列とみなし、音声波形の自己回帰を統計モデル化。
- 音声信号だけでなく、音楽信号を含む音全般の生成に利用可。
- 音質の高さがすごい。

図 8-1 WaveNetに基づく日本語音声合成のフロー



©山本龍一、高道慎之介、Pythonで学ぶ音声合成、2021年8月12日、インプレス

Tactron

・ニューラルネットワークに基づく**一貫学習**を狙った音声合成（2017年8月）。

・従来のパラメトリック音声合成が利用する言語特徴量を介せず、
テキストから振幅スペクトログラムへの変換を単一のニューラルネットで実行。

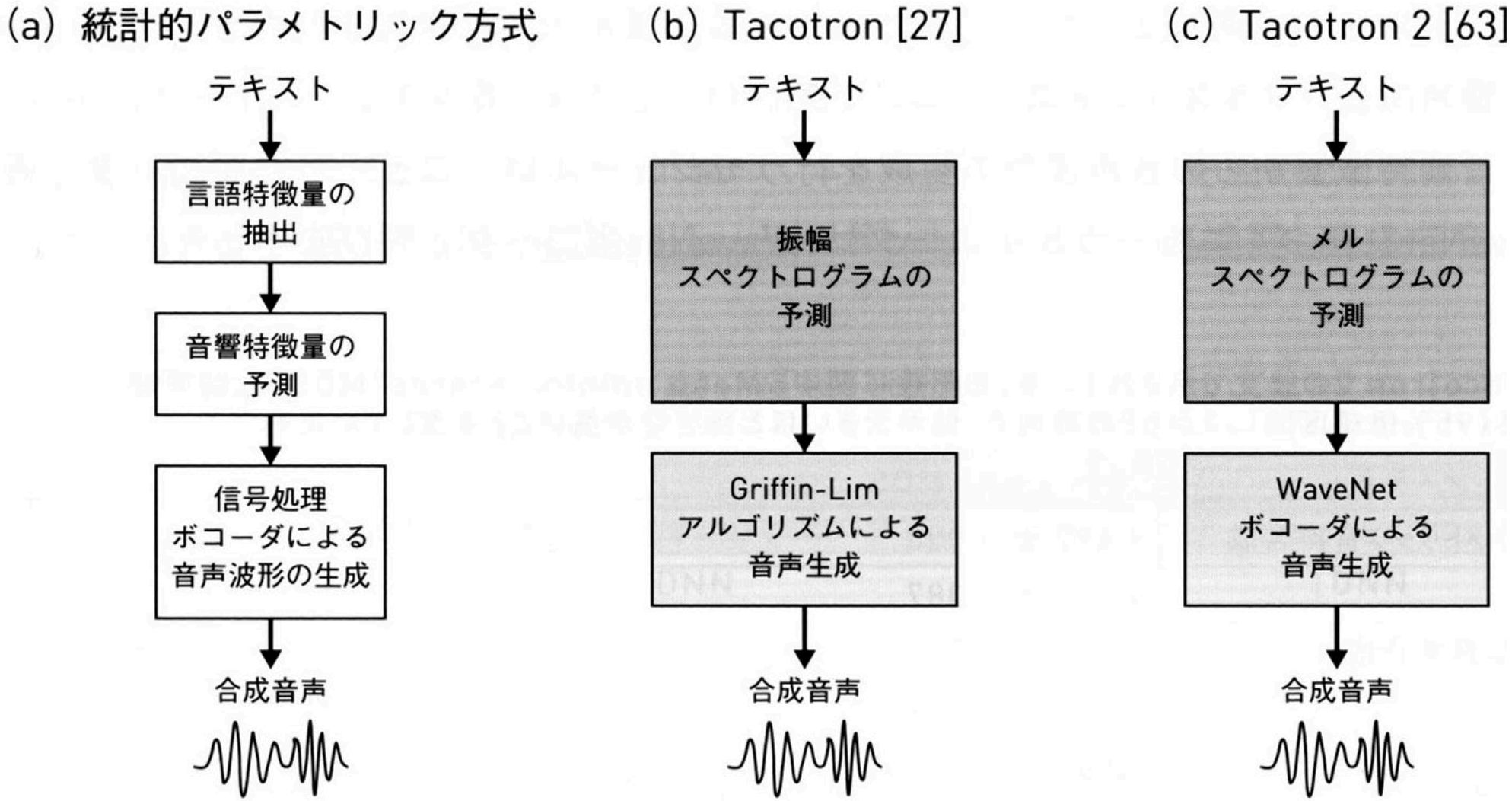
・初期バージョンは素片選択型合成法に劣ったが、
Tactron2（2018年）は自然音声とほとんど同等の品質。

Tactron2 (PyTorch)

Tactron2 (GitHub)

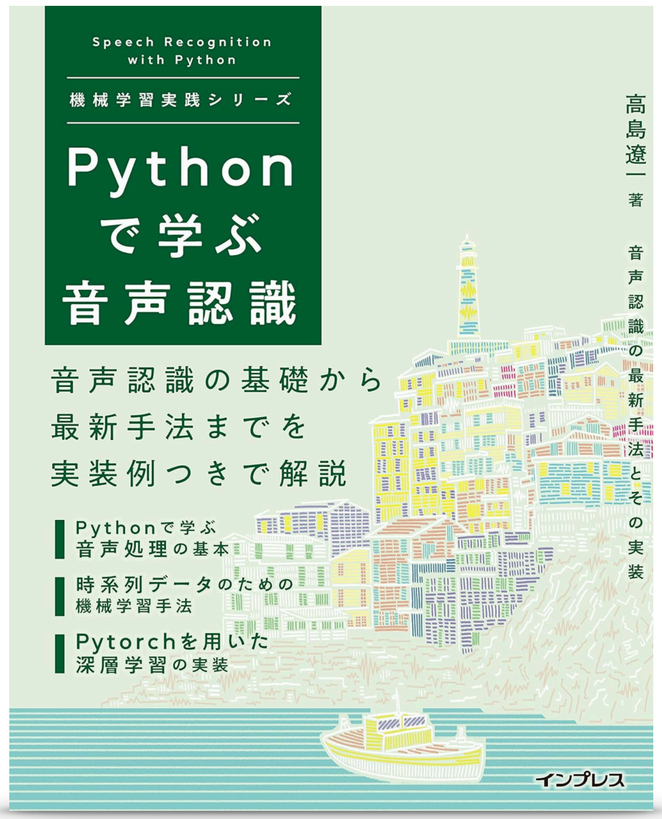
Tactron, Tactron2

図 9-1 従来の統計的パラメトリック音声合成と一貫学習に基づく音声合成の比較



©山本龍一、高道慎之介、Pythonで学ぶ音声合成、2021年8月12日、インプレス

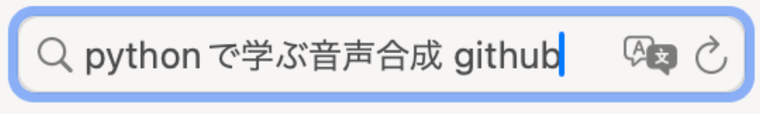
関連専門書

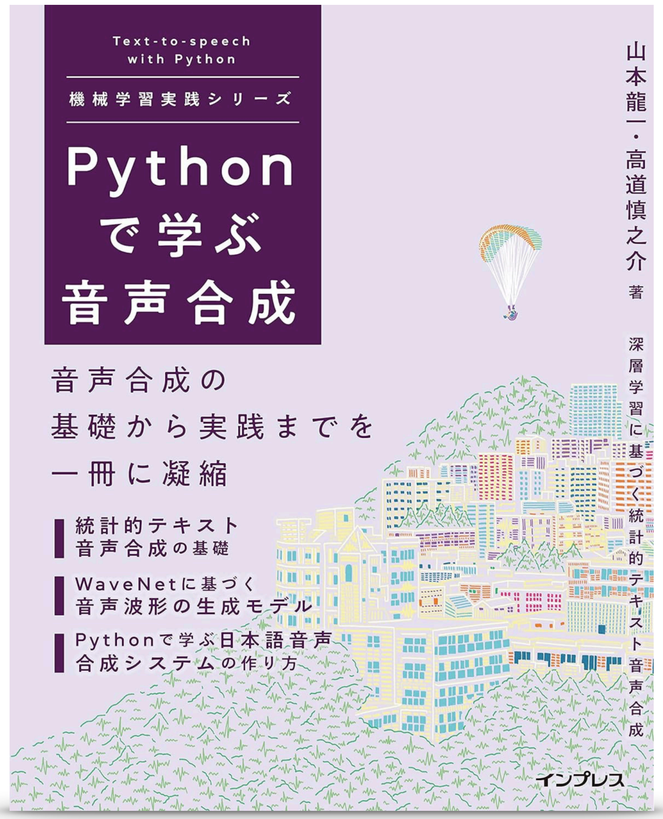


高島遼一

Pythonで学ぶ音声認識（機械学習実践シリーズ）

2021年5月20日、インプレス、￥3,850.





山本龍一、高道慎之介

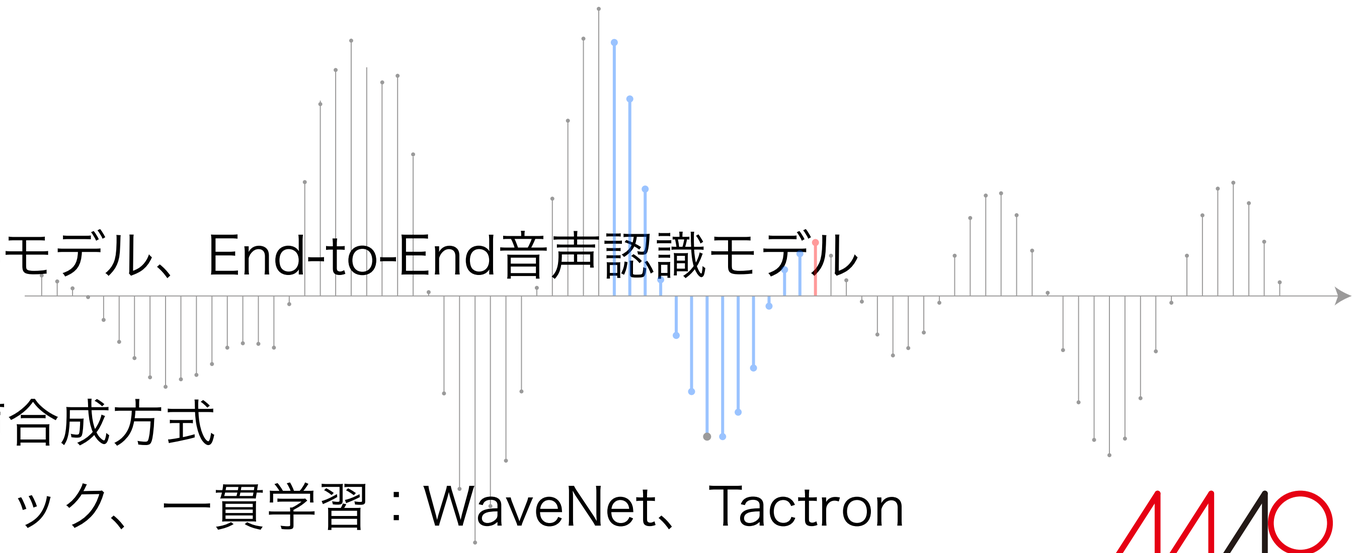
Pythonで学ぶ音声合成（機械学習実践シリーズ）

2021年8月12日、インプレス、￥3,850.

音声のディジタル表現と処理

今日の要点

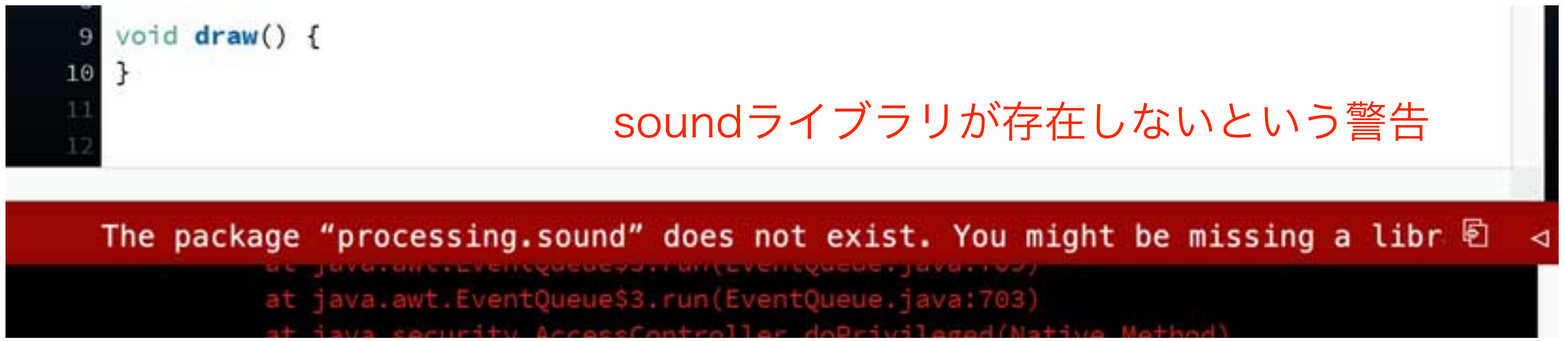
- 音の構成要素：高さ、強さ、音色
- 音の標本化と量子化
  - A/D（Analog to Digital）、D/A（Digital to Analog変換）
- 音声音響データ
  - 波形符号化 → 分析合成符号化 → ハイブリッド符号化
  - MP3, AAC, MMA, AIFF, WAVE, MIDI
- 音声認識
  - スペクトル分析、音響モデル
  - 言語モデル
  - ニューラスネットワーク音響モデル、End-to-End音声認識モデル
- 音声合成
  - 分析合成方式、テキスト音声合成方式
  - 素片編集、統計的パラメトリック、一貫学習：WaveNet、Tactron



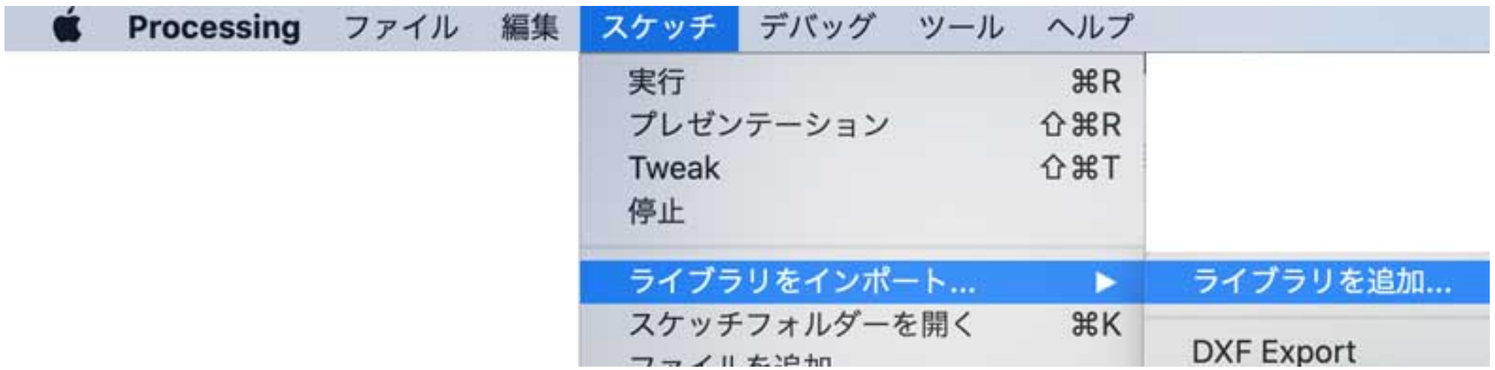
Processingで音を出す

1. soundライブラリをインストールする

Processingインストール直後はsound関連機能が無い。



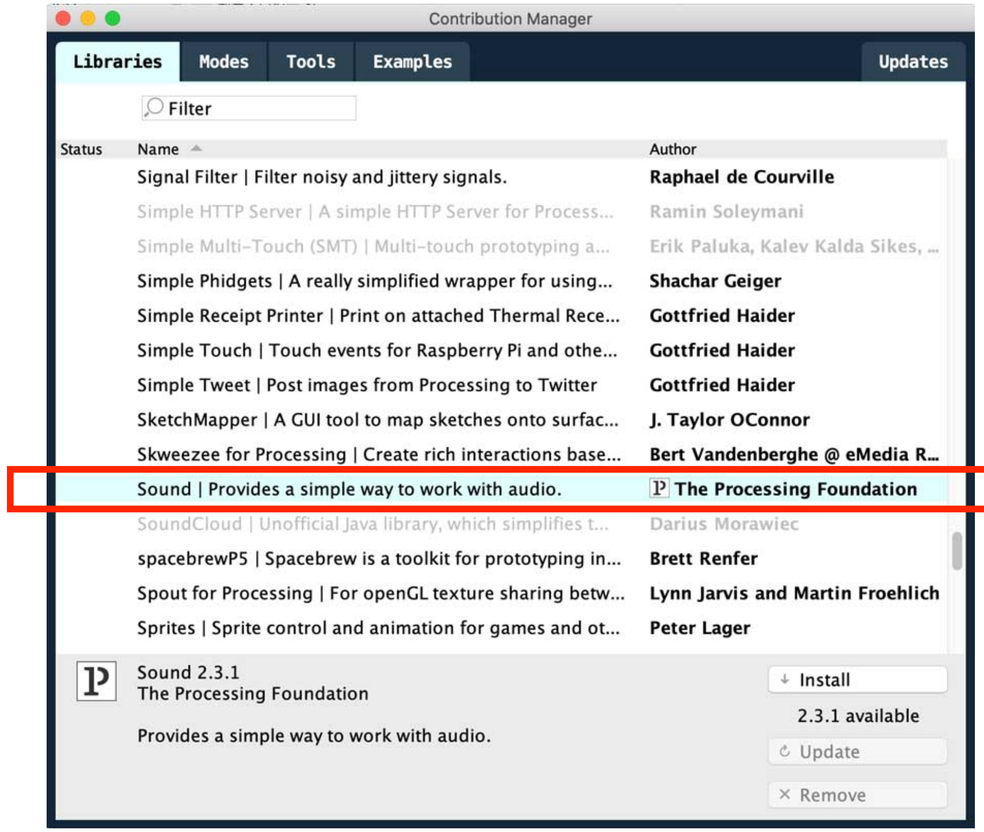
ライブラリを取り込む



Processingで音を出す

1. soundライブラリをインストールする

ライブラリはアルファベット順に並んでいる。

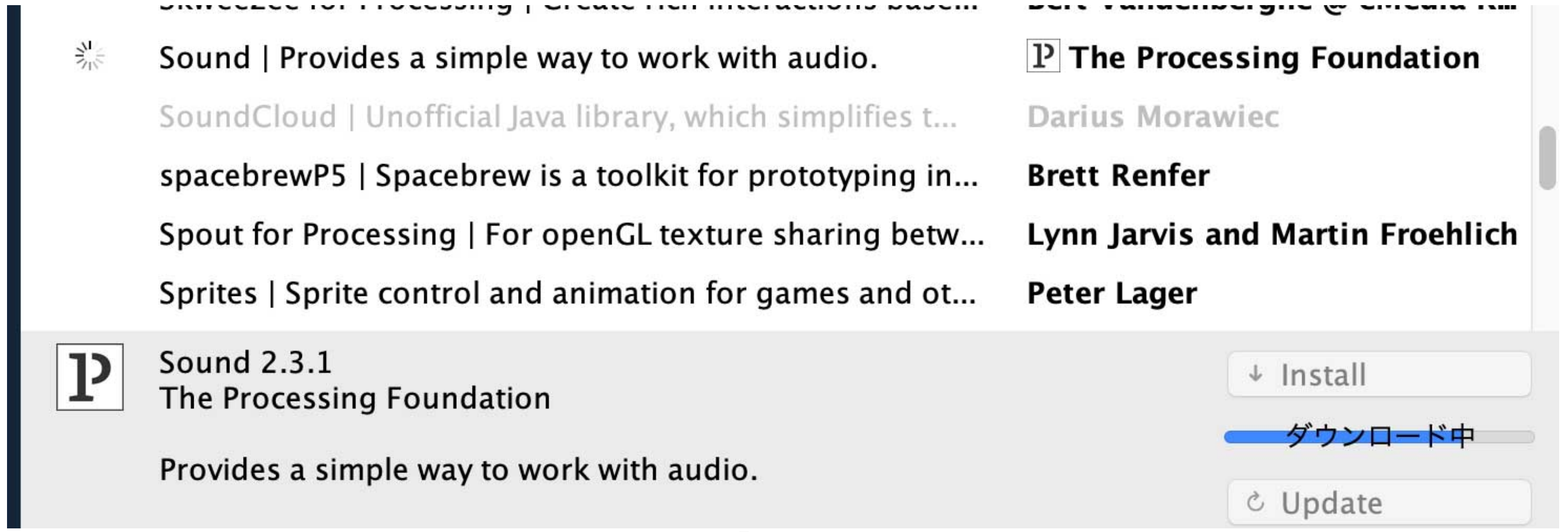


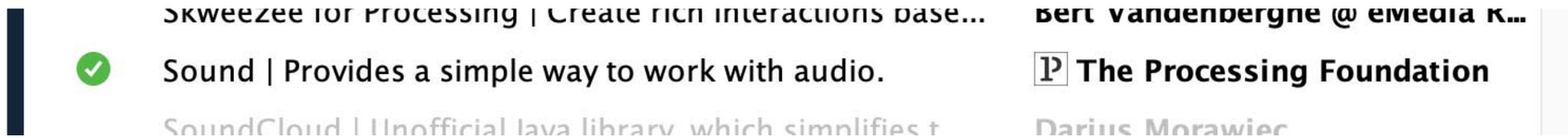
soundライブラリは
Processing.orgが提供

Processingで音を出す

1. soundライブラリをインストールする

↓ Install ボタンを押してインストール





導入完了

Processingで音を出す

2. とりあえず正弦波を鳴らす

```
import processing.sound.*;
SinOsc sine;
void setup() {
  // Create the sine oscillator.
  sine = new SinOsc(this);
  sine.play();
}

void draw() {
}
```

ライブラリを呼び込む：**import** ライブラリ名

SinOsc 正弦波クラス

new：インスタンスを生成

sine.play()：**play**メソッド（音の再生）を実行

https://processing.org/reference/libraries/sound/による

Processingで音を出す

3. 周波数と振幅を指定して正弦波を鳴らす

```
import processing.sound.*;
SinOsc sine;
void setup() {
  // Create the sine oscillator.
  sine = new SinOsc(this);
  sine.freq(1000);
  sine.amp(0.2);
  sine.play();
}

void draw() {
}
```

sine.freq()：freqメソッド（周波数設定）を実行

sine.amp()：ampメソッド（周波数設定）を実行

https://processing.org/reference/libraries/sound/による

Processingで音を出す

3. 周波数と振幅を指定して正弦波を鳴らす（その２）

```
import processing.sound.*;
SinOsc sine;
void setup() {
  // Create the sine oscillator.
  sine = new SinOsc(this);
  sine.play(1000,0.2);
}

void draw() {
}
```

`sine.play()`：周波数と振幅を指定し再生

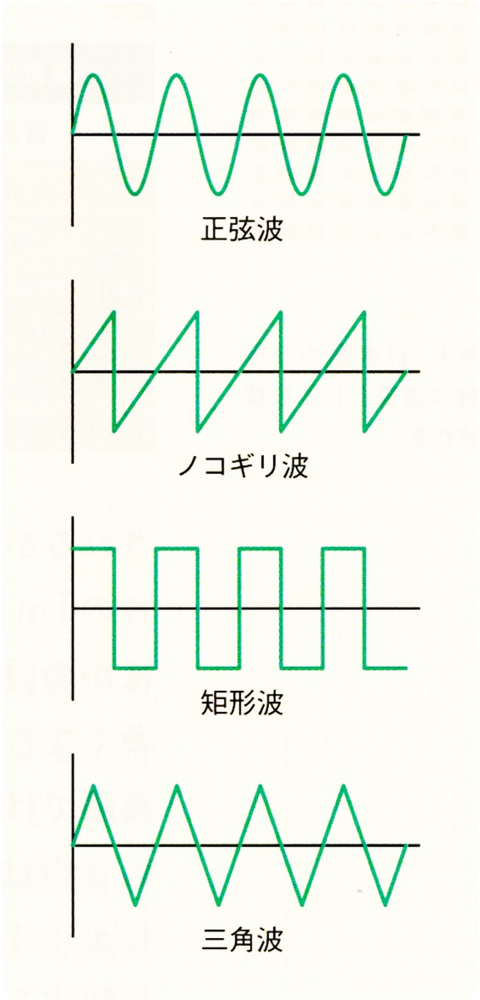
周波数の単位はHz

振幅の値の範囲は0.0〜1.0

https://processing.org/reference/libraries/sound/による

Processingで音を出す

4. 様々な種類の周期波形を鳴らす



■図 1.18——さまざまな音の波形

Processingで音を出す

5. 白色雑音

全ての周波数で同じ強度となるノイズ

```
import processing.sound.*;
WhiteNoise noise;

void setup() {
  // Create the noise generator
  noise = new WhiteNoise(this);
  noise.play(0.4);
}

void draw() {
}
```

`WhiteNoise` 白色雑音クラス

`noise.play(0.4)`：振幅0.4で白色雑音を再生
振幅の値の範囲は0.0〜1.0

Processingで音を出す

6. インタラクティブにする

マウス位置に応じて正弦波の周波数と
背景色を変える

```
import processing.sound.*;
SinOsc sine;
void setup(){
  size( 255, 255 );
  sine = new SinOsc(this);
}

void draw(){
  int r = mouseX;
  int g = mouseY;
  int b = 100;
  background( r, g, b );
  sine.play(200+mouseX+mouseY, 0.4);
}
```

`background(r,g,b)` 背景色をRGB値で指定

正弦波の周波数は（200+マウスのX座標＋マウスのY座標） [Hz]

Processingで音を出す

7. インタラクティブにする　マウスを押した時に音と背景色を変える

mousePressed

```
import processing.sound.*;
SinOsc sine;
void setup(){
  size( 255, 255 );
  sine = new SinOsc(this);
}

void draw(){
}

void mousePressed() {
  int r = mouseX;
  int g = mouseY;
  int b = 100;
  background( r, g, b );
  sine.play(200+mouseX+mouseY,0.2);
}
```

draw()の中身は空

mousePressed()
マウスボタンが押された時に実行する関数

Processingで音を出す

8. インタラクティブにする　440Hzの正弦波を鳴らすボタン

```
void setup() {
  size(400, 400);
  sine = new SinOsc(this);
  background(255, 255, 255);
  fill(255, 0, 0);
  rect(100, 100, 200, 200);
}

void mousePressed() {
if((100<=mouseX)&&(mouseX<=300)
 &&(100<=mouseY)&&(mouseY<=300)) {
    sine.play(440,0.2);
  }
}

void mouseReleased() {
  sine.stop();
}
```

`rect(x, y, w, h)` 矩形を描く

左上角の座標は`(x,y)`、幅は `w`、高さは `h`

`mouseReleased()`

マウスボタンが放された時に実行する関数

`sine.stop()`：正弦波の再生を停める

Processingで音を出す

9. 音響ファイルを読み込み再生する

```
import processing.sound.*;
SoundFile file;

void setup() {
  file = new SoundFile(this,"drums.wav");
  file.play();
}

void draw() {
}
```

`SoundFile` 音響ファイルクラス

`./data/drums.wav` 音響データを読み込み
SoundFileインスタンスを生成する。

`file.play()`： 音響データを再生する。

`drums.wav` はProcessingに付属するサンプルデータである。

Processingで音を出す

10. キーボード入力に反応させる

キーに応じて座標に図形を描く

```
void setup() {
  size(800, 800);
  noStroke();
  smooth();
  background(0);
}

void draw() {
}

void keyPressed() {
  int keyIndex = -1;
  int icase = 1;  // 1: uppercase, 2: lowercase
```

keyPressed

noStroke() 図形の縁を描かない

smooth() anti-aliasing：輪郭や色を滑らかにする

keyPressed()
キーボードのキーが押された時に実行する関数

次頁に続く

Processingで音を出す

10. キーボード入力に反応させる

キーに応じて座標に図形を描く

```
if (key >= 'A' && key <= 'Z') { // uppercase characters
    keyIndex = key - 'A';
    icase = 1;
  } else if (key >= 'a' && key <= 'z') { // lowercase characters
    keyIndex = key - 'a';
    icase = 2;
  }
  if (key == 32) {
     // If key is SPACE key then clear the screen
    background(0);
  } else if ( keyIndex >= 0 && keyIndex <=25 )
```

大文字ならば、"A"(0)を基点とした整数値を`keyIndex`とする

小文字ならば、"a"(0)を基点とした整数値を`keyIndex`とする

スペースキーが押された場合は画面を初期状態に戻す

次頁に続く

Processingで音を出す

10. キーボード入力に反応させる　キーに応じて座標に図形を描く

```
} else if ( keyIndex >= 0 && keyIndex <=25 ) {
  float x = map(keyIndex, 0, 25, 0, width);
  float y = random(800);
  float radius = random(100)+10;
  float r = random(255);
  float g = random(255);
  float b = random(255);
  fill(r, g, b);
  if(icase == 1) {
    ellipse(x, y, radius, radius);
  } else if(icase == 2) {
    rect(x, y, random(100), random(100));
  }
}
}
```

map(v,s1,e1,s2,e2)
vの属する区間**[s1,e1]**を
別の区間**[s2,e2]**に写像する

random(n)
0から**n**までの乱数を生成する

ellipse(x,y,w,h)
(x,y)を中心とする
幅**w**、高さ**h**の楕円を描く