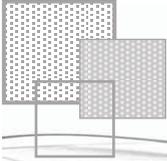


Pythonで学ぶ 実践画像・音声処理入門

博士(工学) 伊藤 克亘
工学博士 花泉 弘 共著
博士(工学) 小泉 悠馬

コロナ社



まえがき

本書の目的は、音声や画像などのデジタルメディアを処理するためのプログラミングの基礎を学ぶことです。デジタルメディア処理の最先端技術の多くは高度な数学に基づいています。それらの技術を習得する第一歩として本書では、理系の大学1, 2年生までに学ぶような数学でもさまざまな処理に役立つことを学びます。その際に、なるべく実際の音声データや画像データを処理して実践的な知識を習得することを目指しています。また、コンピュータによる音声や画像の簡単な加工、分析、生成方法を音声処理と画像処理の共通性を意識しながら学ぶことで、データサイエンスの基礎を学ぶことを目的としています。

対象読者

本書は、Python 3 による Python プログラミングの基本知識を備えていることを想定して執筆しました。具体的には、関数、関数の引数、ループを理解していることを想定しています。

数学に関しては、高校数学および大学レベルの微積分、線形代数、統計学の基礎知識があることが必要ですが、必要に応じて、基礎的なレベルの教科書を参照すれば十分でしょう。またフーリエ変換については、どのようなものか知っている方がサンプルプログラムを理解しやすいでしょう。

本書の構成

本書では、章の最初に、その章のプログラムに必要なパッケージを示します。

説明内容に合わせて、数行のプログラムが示されます。これらのプログラムは、章内では、全部続けて実行することを想定しています。つまり、章の最初の方で値を設定された変数が、後のプログラムで断りなしに使われることもあります。

また、章の最初にキーワードを示します。それに関連する数学的な事項を思い出すとよいでしょう。

その章で学んだことを定着させるための章末問題も用意しています。これらの問題は、学んだ知識を自分なりに応用するためのヒントになっています。定着させるためには、最後まで自力で考えた方がよいのですが、巻末にはヒントを掲載しました。

本書で想定する Python 環境

本書のサンプルプログラムは、Mac OS Sierra 上の Anaconda 4.4.0 をベースとした Python 3.5 の環境で動作確認しています。Windows など異なる環境では、出力などが多少異なるのでご注意ください。パッケージのインストール方法などは、サポートサイトに掲載しています。本書で用いた音声や画像データ、本書のために開発したパッケージもサポートサイト (www.coronasha.co.jp/np/isbn/9784339009026/) からダウンロードできます。

本書で用いるパッケージの一覧を次頁に示します。破線の下は `cis.py` だけで用いるもので、サンプルプログラムなどで直接使うことはありません。

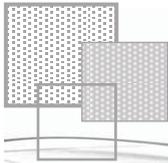
科学技術プログラミングの分野では、本書で紹介した NumPy を基盤とする、さまざまなパッケージが作られています。本書が、そのようなパッケージの利用につながるような音声・画像処理の実践的な Python プログラミングを習得する一歩になれば幸いです。

2018 年 2 月

著 者

本書で用いるパッケージの一覧

パッケージ名	説明
<code>numpy</code>	数学的なアルゴリズムを式に似た形式でプログラミングできる関数
<code>numpy.matlib</code>	プログラミング言語・環境の MATLAB に含まれる便利な関数
<code>numpy.linalg</code>	線形代数
<code>scipy.fftpack</code>	高速フーリエ変換 (FFT)
<code>scipy.io</code>	ファイル入出力
<code>scipy.signal</code>	信号処理
<code>scipy.ndimage</code>	画像処理
<code>scipy.ndimage.filters</code>	画像処理用フィルタ
<code>scipy.spatial</code>	空間処理
<code>scipy.stats</code>	統計処理
<code>skimage.transform</code>	画像の幾何学的変換
<code>skimage.feature</code>	画像の特徴抽出
<code>skimage.util</code>	画像処理ツール
<code>sklearn.cluster</code>	クラスタリング
<code>sklearn.neighbors</code>	機械学習の最近傍法
<code>matplotlib.pyplot</code>	MATLAB のようなグラフ描画
<code>matplotlib.mlab</code>	MATLAB と同一の名称を持つプロット関連の互換関数
<code>cv2</code>	画像処理ライブラリ OpenCV の利用
<code>cis</code>	本書のために作成
<code>simpleaudio</code>	音声入出力
<code>mpl_toolkits</code>	<code>matplotlib</code> 用ツール
<code>plotly</code>	データ可視化



目 次

1 簡単な音声処理

1.1	波形データの生成	1
1.2	1次元データの可視化	5
1.3	時間波形の重ね合わせ	9
1.4	時間波形の連結	12
1.5	読み込んだ音声データの加工	13
	章末問題	15

2 簡単な画像処理

2.1	画像の構造	17
2.2	画像・ビデオの読み込み	21
2.3	領域の抽出	27
	章末問題	32

3 音声のフーリエ変換

3.1	フーリエ変換	34
3.2	窓関数	38
3.3	音声のフレーム処理	41

3.4 逆フーリエ変換	44
章末問題	45

4 フィルタ (音声)

4.1 線形フィルタ	48
4.1.1 線形システム	48
4.1.2 遅延演算	49
4.1.3 移動平均フィルタ	51
4.2 インパルス応答	53
4.3 IIR フィルタ	56
4.4 フィルタ設計のツール	57
章末問題	58

5 画像の周波数領域処理

5.1 空間周波数	60
5.2 2次元フーリエ変換	62
5.3 周波数領域でのフィルタ処理	64
5.4 周波数領域での画像の拡大	68
章末問題	69

6 画像の空間領域処理

6.1 2次元畳み込み	71
6.2 微分演算	74
6.3 エッジの検出	75

6.4 非線形フィルタ	79
章 末 問 題	79

7 音声データの相関

7.1 相互相関	81
7.1.1 ベクトルの類似度	81
7.1.2 相互相関関数	84
7.2 自己相関	86
7.3 時間波形のフレーム処理	88
章 末 問 題	93

8 画像データの類似度

8.1 画素のユークリッド距離	95
8.2 画素の相関の応用	96
8.3 領域の相関	97
章 末 問 題	102

9 複素信号

9.1 信号の複素指数関数表現	104
9.2 周波数変調	106
9.2.1 瞬時周波数	106
9.2.2 周波数変調	107
9.2.3 任意の音の周波数変調	108
章 末 問 題	112

10

画像の幾何学的処理

10.1	2次元平面上の回転	115
10.2	2次元平面上の平行移動	118
10.3	同次座標表現を用いた変換	118
10.4	アフィン変換	120
10.5	射影変換	124
10.6	複雑な形状の変換	124
	章末問題	127

11

分類

11.1	特徴量	130
11.1.1	短時間エネルギー	131
11.1.2	零交差	132
11.2	k最近傍分類	134
11.3	最尤法	138
	章末問題	140

12

音声・画像処理の応用

12.1	Wavetable合成	142
12.1.1	ADSRエンベロープ	142
12.1.2	楽器音からの波形データの抽出	145
12.1.3	複数のテンプレートを用いた合成	146
12.1.4	長さの変更	149

12.1.5	リサンプルによるピッチの変更	150
12.2	衛星画像の時間変化領域の解析	150
章 末 問 題		154
章末問題ヒント		158
索 引		175

1

簡単な音声処理

まず、1次元のデジタルデータの代表として音データを取り上げる。この章ではコンピュータに読み込んだ音声データを用いて、本書で対象とする中心パッケージである `numpy` と `matplotlib` の基本機能を習得する。

— 利用するパッケージ —

```
import numpy as np
import matplotlib.pyplot as plt
import cis
```

キーワード 振幅, 周波数, サンプル周期, サンプル周波数, 離散的, スカラ, ベクトル, 可視化

1.1 波形データの生成

最も単純な音の一つに純音がある。純音は正弦波で表される。物理の教科書では、純音は式 (1.1) のように表される。

$$y = A \sin(2\pi ft) \tag{1.1}$$

ここで、 y は音圧、 A は振幅、 f は周波数、 t は時間である。時間を横軸にとった y の変化のグラフが図 1.1 である。

グラフは2次元であるが、 y という変数が各時刻で一つの値をとり、その値が時間で変化するので1次元データと呼ぶ。

式 (1.1) に基づいて、Python で音のデジタルデータを生成するプログラムを作成する。データを生成するためには、まず、式 (1.1) の変数の値を決めなけ

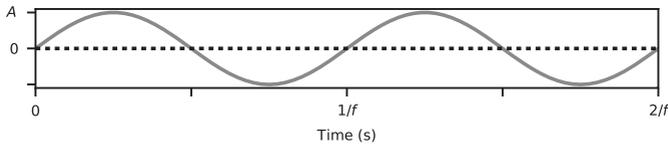


図 1.1 正弦波のグラフ

ればならない。 A , f は、それぞれ一つの値を定めればよい。例えば、 $A = 1$, $f = 50$ などである。このように大きさのみを持つ量のことをスカラと呼ぶ。

一方、 t については、例えば、1 秒間のデータを作成するときには、開始時刻を 0 秒とすると、時刻 0 秒から 1 秒まで変化する。式で書くと $0 \leq t \leq 1$ となる。つまり一つの値ではない。

物理的な世界では、 t は連続的に変化する。このような連続的な量をコンピュータで扱う最も一般的な方法は、均等で微小な間隔の値の列として表現することである。例えば、 $1/8000$ 秒の間隔とすると、 $0 \leq t \leq 1$ という範囲は、 $0, 1/8000, 2/8000, 3/8000, \dots, 7999/8000, 1$ という数の列で表される。この $1/8000$ 秒をサンプリング周期と呼ぶ。また、この周期に対応する周波数をサンプリング周波数と呼ぶ。周波数は周期の逆数なので、この場合は 8000 Hz となる。

このようにとびとびの値で表すことを離散的であるという。つまり、コンピュータの中では正弦波は、図 1.2 の丸印のところだけで表現されている。

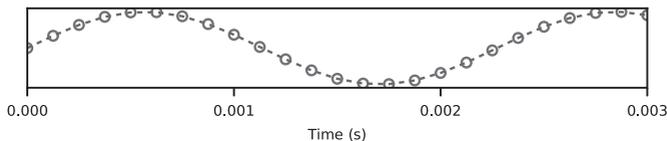


図 1.2 離散的な音声データ

このようなデータをプログラミングするときには、 t をひとまとまりで扱うと便利である。そのような場合、数学的にはベクトルとして扱うことがある。プログラムでは、ベクトルをそのままの形で扱えると便利である。

● 正弦波の生成 ベクトルをそのまま扱える `numpy` パッケージを利用して 440 Hz の音を 1 秒間生成してみる。プログラム 1-1 のように入力する（先頭に数字がある場合は行番号を表す。また、`>>>`は、Python が表示するプロンプトである。したがってどちらも入力しなくてよい）。

——— プログラム 1-1（正弦波の生成） ———

```
1 >>> t=np.arange(0,1,1/8000)
2 >>> a=0.8
3 >>> f=440
4 >>> y=a*np.sin(2*np.pi*f*t)
```

このように、ベクトルを用いた計算を、非常に簡単に、ほとんど数式と同じ形で記述するだけでプログラミングできる。1 行目では、時間を表す数列を作成している。`np.arange` の `np.` の部分は、章の冒頭で示した `numpy` というパッケージであることを示している。つまり、`np.arange` とは `numpy` パッケージの `arange` という関数であることを意味する。`arange` の使い方を示す。

```
>>> np.arange(0,10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

`arange` は、このように、数列を作成する関数である。第 1 引数は、数列の開始値であり、第 2 引数の一つ前の値で終了するように指定する。この例の場合、0 から 10 の一つ前の 9 までの整数列が生成され、`n` に代入される返り値には、データ型が `array` であることが明示されている。

プログラム 1-1 の `arange` の例では引数が三つある。この場合、第 3 引数が生成する数列の間隔となる。間隔を指定する例を示す。

```
>>> np.arange(-1,2,0.5)
array([-1. , -0.5,  0. ,  0.5,  1. ,  1.5])
```

指定した通りに 0.5 刻み間隔の数列が生成される。第 2 引数は 2 なので、2 の一つ前の 1.5 が最後の要素となる。

プログラム 1-1 の 2 行目は、振幅を設定している。ここでは、振幅が指定さ

4 1. 簡単な音声処理

れていないので、適当な値としている。4行目は、式(1.1)をNumPyでプログラミングしている。数式とほとんど同じ形で表現できることに注目してほしい。この `pi` は、NumPy であらかじめ用意されている変数で π の値を持つ。

`sin` は、三角関数の `sin` である。NumPy で用意されている関数については、`info` 関数で説明を見ることができる。

```
>>> np.info(np.sin)
sin(x[, out])

Trigonometric sine, element-wise.

Parameters
-----
x : array_like
(以下略)
```

この説明に `x : array_like` とあるように、`sin` 関数は引数に配列（数列）をとれる。引数に数列をとった場合の例を示す。

```
>>> np.set_printoptions(precision=3)
>>> y2=np.sin(np.arange(0,1,0.1))
>>> y2
array([ 0. ,  0.1 ,  0.199,  0.296,  0.389,  0.479,  0.565,  0.644,
        0.717,  0.783])
```

1行目 `set_printoptions` は出力を制御する関数である。ここでは、最長で小数点以下3桁になるように抑制している。この例では、`sin` の引数は数列である。その場合、計算結果の `y2` も数列となり、その値は引数の `sin` の値である。このようにNumPyの多くの関数は数列や行列を引数にとることができる。この機能を活用すると、NumPyを用いて数式とほぼ同じ形でプログラミングできるようになる。

プログラム1-1の4行目では、`t` は数列である。`t` に掛けられている `2*np.pi*f` は、この場合 `f` は 440 なので、 $2 \times 3.14 \times 440 = 2763.2$ となりスカラである。NumPy では、ベクトルにスカラが掛けられている場合はベクトルのそれぞれ

の要素をスカラ倍する。したがって、 $2 * \text{np.pi} * f * t$ は、数列 t のおのおのの要素を $2\pi f$ 倍する。すべての要素が $2\pi f$ 倍された数列に対して \sin を計算し、その結果の数列を a 倍している。

作成した音データを出力するための関数が `audioplay` である。プログラム 1-1 で作成した y はつぎのようにして出力する。

```
>>> cis.audioplay(y,8000)
```

第1引数が出力したい数列、第2引数はサンプリング周波数である。周波数が 440 Hz の「ラ」の音が1秒間聞こえるはずである。

1.2 1次元データの可視化

音を生成したり、加工したり、録音した場合には、もちろん、音を再生、出力して確認すべきである。しかし、音は聞こえ方が人によってかなり異なるし、プログラムに失敗していたら、聞こえる音にならなかったり、デバイスに悪影響を与えるようなデータになることもある。したがって、聞く以外の方法でも確認した方がよい。普通には見ることができないデータを見えるようにすることを可視化という。

まず、時間に対する音圧の変化のグラフで確認する方法を取り上げる（音声波形、時間波形のプロットと呼ばれることが多い）。

```
>>> plt.plot(y)
[<matplotlib.lines.Line2D object at 0x110d515c0>]
>>> plt.show()
```

1行目の `plot` は、数列をプロットする関数である。特に指定しなければ、横軸を数列のインデクス（番号）とし、縦軸を数列の値として、直線をつないでプロットする。ただし、この段階では、グラフは表示されず、システムの出力が返される（2行目、`at`以降の16進数は実行環境により変化するため、この例とは異なった値となる）。3行目の `show` によりグラフが表示される（図 1.3）。

2

簡単な画像処理

2次元、もしくは多次元のデジタルデータの代表として画像データを取り上げる。この章では、コンピュータに読み込んだ画像データを用いて、画像データの構造や基本的な扱いを説明する。

—— 利用するパッケージ ——

```
import numpy as np
import matplotlib.pyplot as plt
import numpy.matlib as mlb
import cv2
import cis
```

キーワード 座標, 画素, RGB, バンド

2.1 画像の構造

[1] グレイスケール画像 画像も音と同じようにコンピュータの中では数字の組で表現される。まず、簡単な例として、グレイスケール画像（白黒画像）の例を見てみる。

```
1 >>> x=np.linspace(255,0,12)
2 >>> x
3 array([ 255. , 231.81818182, 208.63636364, 185.45454545,
4         162.27272727, 139.09090909, 115.90909091, 92.72727273,
5         69.54545455, 46.36363636, 23.18181818, 0. ])
6 >>> x=x.astype(np.uint8)
7 >>> x
8 array([255, 231, 208, 185, 162, 139, 115, 92, 69, 46, 23, 0],
```

```

9 dtype=uint8)
10 >>> x=x.reshape(3,4)
11 >>> x
12 array([[255, 231, 208, 185],
13        [162, 139, 115, 92],
14        [ 69, 46, 23, 0]], dtype=uint8)
15 >>> plt.imshow(x,cmap='gray')
16 <matplotlib.image.AxesImage object at 0x116c1b208>
17 >>> plt.show()

```

1 行目の `linspace` は、開始値と終了値を指定して、その間に指定した個数の等間隔の点からなる配列（ベクトル）を生成する。ここでは、255 から 0 までの 12 点を生成している。6 行目の `astype` は配列の型を変更するメソッドである。ここでは、元の `float` 型の値を 8 ビットの符号なしの整数型（`uint8`, 0～255 の範囲の整数）に変換している。10 行目の `reshape` は配列の形状を変更して出力するメソッドである。それを用いて配列を 3×4 の 2 次元配列に整形している。11 行目の出力結果からわかるように、2 次元の場合は、行方向に順に並ぶように整形される。15 行目で画像を描画しているが、17 行目を実行するまで表示されない（図 2.1）。

この画像は、12 個の画素から構成される。実際の画素は非常に小さいので、ディスプレイ上では、実寸で表示すると見えないくらい小さい。しかし、`imshow` は画像を適当に拡大、縮小して描画する関数である。この例の場合は、実際の画

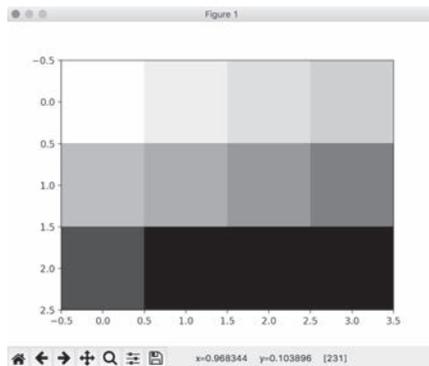


図 2.1 画素の情報表示

素よりはるかに大きく表示される。実際には、左上隅の白い部分は画素一つ分で、その座標は (0,0) であり、その下は (0,1), 右は (1,0) となる。imshow では、マウス位置の座標とその場所の画素の値がウィンドウ下部のメニューバーの右側に表示される。図 2.1 は (1,0) の部分を表示させたところで、画素の値が 231 と表示されている。この例の画像はグレイスケール画像と呼ばれ、画素の値が大きいほどその画素は明るくなり、小さいほど暗くなる。ここでは、描画するときに、cmap 引数を gray とすることでグレイスケール画像として描画されている。

画像の大きさは、横方向の画素数を幅 (width) と呼び、縦方向の画素数を高さ (height) と呼ぶ。図 2.1 では、幅が 4 で高さが 3 である。

[2] RGB 画像 RGB と呼ばれるカラー画像では、画素は赤、緑、青 (red, green, blue, 略して RGB) の三つの値の組 (ベクトル) で表現される。プログラム 2-1 に色の指定方法を示す。

プログラム 2-1 (さまざまな色の生成)

```

1 bar0=np.zeros((500,100),np.uint8)
2 bar1=255*np.ones((500,100),np.uint8)
3 Col=np.zeros((500,800,3),np.uint8)
4 Col[:, :, 0]=mlb.repmat(np.hstack((mlb.repmat(bar1,1,2),
5 mlb.repmat(bar0,1,2))),1,2)
6 Col[:, :, 1]=np.hstack((mlb.repmat(bar1,1,4),mlb.repmat(bar0,1,4)))
7 Col[:, :, 2]=mlb.repmat(np.hstack((bar1,bar0)),1,4)
8 plt.imshow(Col)
9 plt.show()

```

左から白、黄、シアン、緑、マゼンタ、赤、青、黒の帯が生成される。NumPy では、Col[:, :, 1] のように、三つの添え字を持つ配列で 3 次元配列を表す。この例の場合、Col は、500 × 800 の 2 次元配列三つから構成される。それぞれの 2 次元配列は、プレーン、バンドなどと呼ばれる。

NumPy では、この例のように配列の要素を指定する部分に「:」だけを書いた場合は、開始値 0 と終了値が最後の要素までという指定を省略したことになるのですべての要素を指定したことになる。RGB 画像では、3 次元配列の最初

のバンドが赤、つぎが緑、最後が青に対応する。そのことは、表示された画像の赤の部分でマウスボタンを押しながら少し動かしてみて、画素の値を表示させると [255,0,0] と表示されることでわかる。

1 行目の `zeros`、2 行目の `ones` は配列を 0 や 1 で初期化するために使われる。第 1 引数は、作成したい配列の大きさをタプルで指定する。(500,100) は 2 次元配列で第 1 次元が 500 個、第 2 次元が 100 個、行列としては 500×100 であることを示す。第 2 引数は配列の (数) 値の型を指定する。ここでは符号なし 8 ビット整数である。`ones`、`zeros` の使い方をつぎに示す。

```
>>> np.ones((2))
array([ 1.,  1.])
>>> np.zeros((2,2),np.uint8)
array([[0, 0],
       [0, 0]], dtype=uint8)
```

プログラム 2-1 の 4 行目の `repmat` は、行列を繰り返して大きな行列を作成する関数である。

```
1 >>> x1=np.array([[1],[2]])
2 >>> x1
3 array([[1],
4        [2]])
5 >>> x1.shape
6 (2, 1)
7 >>> mlb.repmat(x1,1,2)
8 array([[1, 1],
9        [2, 2]])
10 >>> mlb.repmat(x1,2,1)
11 array([[1],
12        [2],
13        [1],
14        [2]])
15 >>> mlb.repmat(x1,2,2)
16 array([[1, 1],
17        [2, 2],
18        [1, 1],
19        [2, 2]])
```

上記 1 行目の `array` は `numpy` の多次元配列を作成する関数である。ここで作成している `x1` は 2 行 1 列の行列である。`repmat` は行列を第 1 引数に指定し、その行列を行方向（縦）に第 2 引数の回数、列方向（横）に第 3 引数の回数だけ繰り返した行列を作成する。7 行目では列ベクトルの `x1` を行方向に 1 回、列方向に 2 回、つまり列方向に 2 回繰り返して 2×2 の行列を作成している。

プログラム 2-1 の 6 行目の `hstack` は、ベクトルだけでなく行列も横に並べることができる。つぎの例の 6 行目が行列を並べる例である。

```
1 >>> x2=np.array([[3],[4]])
2 >>> x12=np.hstack((x1,x2))
3 >>> x12
4 array([[1, 3],
5        [2, 4]])
6 >>> np.hstack((x12,x12))
7 array([[1, 3, 1, 3],
8        [2, 4, 2, 4]])
```

2.2 画像・ビデオの読み込み

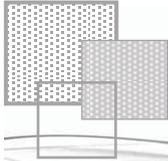
[1] 画像の読み込み Python では、つぎのように OpenCV の関数を用いて画像を読み込める。

```
1 >>> I=cv2.imread('paprika-966290_640.jpg')
2 >>> plt.imshow(cv2.cvtColor(I, cv2.COLOR_BGR2RGB))
3 <matplotlib.image.AxesImage object at 0x1141b3438>
4 >>> plt.show()
```

OpenCV の関数でカラー画像を読み込むと、プログラム 2-1 と違って、バンドが青、緑、赤の順で並べられる。このような画像を BGR 画像と呼ぶ。2 行目では、`cvtColor` 関数で BGR から RGB に変換している。

この画像の大きさは、`shape` メソッドで調べられる。

```
1 >>> sz=I.shape
```



索引

【あ・い】		【く】		縮小	119
アフイン変換	121	空間周波数	62	出力	48
位相	37	空間周波数スペクトル	61	純音	1
位相角	37	クラスタリング	141	瞬時周波数	107
位相平面	37	グレイスケール画像	17	振幅	1
1次式	49	【さ】		振幅スペクトル	35
移動平均フィルタ	51, 79	斉次座標表現	118	振幅変調	12
インデックス	5, 7	最短距離法	141	【す】	
インパルス	53	最尤法(分類)	139	スカラ	2
インパルス応答	53	雑音	51	スペクトル	35
【え・お】		座標	19	スペクトログラム	43
衛星画像	136	散布図	98	【せ】	
エッジ	61, 96	サンプリング周期	2, 49	正規化	15
エッジ検出	77	サンプリング周波数	2	正規化相関法	141
オイラーの公式	104	【し】		正弦波	1
音声波形	5	時間解像度	44	零交差	132
【か】		時間波形	5	鮮鋭化	77
回転	115	時間領域	44	漸化式	56
拡大	119	しきい値	28	線形システム	49
角度	82	自己相関関数	87	線形チャープ	107
加算	48	システム	48	せん断	128
可視化	5	射影変換	124	【そ】	
画素	18	周期関数	86	相関係数	84
カラーエッジ	96	周期性	37	相関係数行列	97
【き】		周波数	1	相互相関関数	85
基本周波数	87	周波数応答	54	【た】	
逆フーリエ変換	44	周波数成分	35	ダイナミックレンジ	98
教師付き分類	135	周波数特性	54	多次元正規分布	139
距離	81	周波数分解能	36	畳み込み	53
		周波数変調	107	多バンド画像	151
		周波数領域	44		

単位インパルス	53	フレーム (音声)	42	BGR	21
短時間エネルギー	131	フレーム (ビデオ)	24		
		フレームシフト	89		
【ち・て】		分散	138	【C】	
遅延	49	分類	130	ceil	146
チャープ信号	106			clip	26
中央値	79	【へ・ほ】		:	8
直流成分	44, 62	平均値	138	convolve	53, 73
定数倍	49	平行移動	118	convolve2d	71
デローネイ三角分割	124	ベクトル	2	corrcoef	97
		偏角	37	correlate	85
【と】		変換行列	116	correlated2d	100
同次座標表現	118	補間	109	cvtColor	21
特徴空間	95, 139	ホワイトノイズ	51		
特徴量	131			【D・E】	
		【ま・め・ゆ】		Delaunay	125
【な・に・の】		マスク	27	dot	82
ナイキスト周波数	36	窓関数	39	exec	10
内積	82	メディアン	79	exp	105
入力	48	ユークリッド距離	81, 95		
ノルム	82			【F】	
		【ら・り】		f ₀	87
【は】		ラブラシアンオペレータ	78	FFT	35
バイナリマスク	27	リーク	38	fft	34
白色雑音	51	離散的	2	fftconvolve	76
外れ値	98	離散フーリエ変換	34	fftshift	44
ハニング窓	39			fft2	64
反転	128	【る・れ・ろ】		figure	44
バンド	19	類似度	81	FIR	56
ハン窓	39	零交差	131	firwin	57
		ローパスフィルタ	57		
【ひ】		論理インデクス	28	【G・H】	
非線形フィルタ	79			getAffineTransform	123
ビブラート	107	【A・B】		hanning	39
微分係数	74	@	83	HPF	59, 65
標準正規分布	52	ADSR エンベロープ	143	hstack	12
		angle	37	hypot	77
【ふ】		arange	3		
フィードバック	56	argrelmax	91	【I】	
フィルタ	50	argwhere	110	ifft	44
複素指数関数	104	astype	18	ifft2	66
複素平面	37	audioplay	5	IIR	56
フーリエ変換	34	axis	116	implay	25
				imshow(cv2)	25

—— 著者略歴 ——

伊藤 克亘 (いとう かつのぶ)

- 1993 年 東京工業大学大学院理工学研究科博士課程修了 (情報工学専攻)
博士 (工学)
- 1993 年 電子技術総合研究所研究員
- 2003 年 名古屋大学大学院助教授
- 2006 年 法政大学教授
現在に至る

花泉 弘 (はないずみ ひろし)

- 1981 年 東京大学大学院工学系研究科博士課程中退 (計数工学専攻)
- 1981 年 東京大学助手
- 1987 年 工学博士 (東京大学)
- 1987 年 法政大学専任講師
- 1989 年 法政大学助教授
- 1996 年 法政大学教授
現在に至る

小泉 悠馬 (こいずみ ゆうま)

- 2014 年 法政大学大学院情報科学研究科博士前期課程修了 (情報科学専攻)
- 2014 年 日本電信電話株式会社 NTT メディアインテリジェンス研究所研究員
現在に至る
- 2017 年 電気通信大学大学院情報理工学研究科博士後期課程修了 (情報学専攻)
博士 (工学)

Python で学ぶ実践画像・音声処理入門

Introduction to Media Computing in Python : A Practical Approach

© Katsunobu Ito, Hiroshi Hanaizumi, Yuma Koizumi 2018

2018年4月27日 初版第1刷発行



検印省略

著者 伊藤 克 亘
花 泉 弘
小 泉 悠 馬
発行者 株式会社 コロナ社
代表者 牛来 真也
印刷所 三美印刷株式会社
製本所 有限会社 愛千製本所

112-0011 東京都文京区千石 4-46-10
発行所 株式会社 コロナ社
CORONA PUBLISHING CO., LTD.
Tokyo Japan

振替 00140-8-14844・電話 (03)3941-3131 (代)

ホームページ <http://www.coronasha.co.jp>

ISBN 978-4-339-00902-6 C3055 Printed in Japan

(三上)



JCOPY <出版者著作権管理機構 委託出版物>

本書の無断複製は著作権法上での例外を除き禁じられています。複製される場合は、そのつど事前に、出版者著作権管理機構（電話 03-3513-6969, FAX 03-3513-6979, e-mail: info@jcopy.or.jp）の許諾を得てください。

本書のコピー、スキャン、デジタル化等の無断複製・転載は著作権法上での例外を除き禁じられています。購入者以外の第三者による本書の電子データ化及び電子書籍化は、いかなる場合も認めていません。落丁・乱丁はお取替えいたします。