画像のウェーブレット解析の試み

岡本安晴

2023.10

ウェーブレット解析が金谷（2003；36 刷 2023）に分かり易く説明されていた。これは 1 変数の関数が扱われているが、２変数の関数（画像データ）の場合を、Nason (2008, sec. 2.13)を参考にして、以下のように考えた。

２変量の関数$f^{(0)}(x,y)$において、領域$[0,2^n)\times[0,2^n)$の外では 0 であるものについて考える。さらに、各矩形領域の中では定数であるとする。すなわち、

$$f^{(0)}(x,y)=C^{(0)}_{h,k},\qquad h\le x<h+1, k\le y<k+1$$

である。

いま、スケーリング関数ϕとウェーブレット母関数ψを次のように置く。

$$\phi(x)=\begin{cases}1, & 0\le x<1\\ 0, & \text{その他}\end{cases}$$

$$\psi(x)=\begin{cases}1, & 0\le x<1/2\\ -1, & 1/2\le x<1\\ 0, & \text{その他}\end{cases}$$

このとき、さらに次のように置く。

$$\phi^{(j)}_h(x)=\phi\big((x-2^jh)/2^j\big)$$
$$\psi^{(j)}_k(x)=\psi\big((x-2^jk)/2^j\big)$$
$$\Phi^{(j)}_{h,k}(x,y)=\phi^{(j)}_h(x)\phi^{(j)}_h(y)$$
$$\Psi^{(j)}_{H,h,k}(x,y)=\psi^{(j)}_h(x)\phi^{(j)}_h(y)$$
$$\Psi^{(j)}_{V,h,k}(x,y)=\phi^{(j)}_h(x)\psi^{(j)}_h(y)$$
$$\Psi^{(j)}_{D,h,k}(x,y)=\psi^{(j)}_h(x)\psi^{(j)}_h(y)$$

これらの関数$\Psi^{(j)}_{H,h,k}$、$\Psi^{(j)}_{H,h,k}(x,y)$、$\Psi^{(j)}_{V,h,k}$、$\Psi^{(j)}_{D,h,k}$は互いに直交している。したがって、ノルムを 1 に調整した以下の関数は正規直交系を構成する。

$$\Phi^{(j,0)}_{h,k}(x,y)=\frac{1}{2^j}\Phi^{(j)}_{h,k}(x,y)$$

$$\Psi^{(j,0)}_{H,h,k}(x,y)=\frac{1}{2^j}\Psi^{(j)}_{H,h,k}(x,y)$$

$$\Psi^{(j,0)}_{V,h,k}(x,y)=\frac{1}{2^j}\Psi^{(j)}_{V,h,k}(x,y)$$

$$\Psi^{(j,0)}_{D,h,k}(x,y)=\frac{1}{2^j}\Psi^{(j)}_{D,h,k}(x,y)$$

いま、

$$V^{(0)} = \left\{\Phi_{h,k}^{(0,0)}\right\};\quad \Phi_{h,k}^{(0,0)}\text{で張られる空間}$$

と置けば、

$$f^{(0)} \in V^{(0)}$$

である。

さらに、

$$V^{(j)} = \left\{\Phi_{h,k}^{(j,0)}\right\}$$

$$W_H^{(j)} = \left\{\Psi_{H,h,k}^{(j,0)}\right\}$$

$$W_V^{(j)} = \left\{\Psi_{V,h,k}^{(j,0)}\right\}$$

$$W_D^{(j)} = \left\{\Psi_{D,h,k}^{(j,0)}\right\}$$

と置けば、

$$V^{(j-1)} = V^{(j)} \oplus W_H^{(j)} \oplus W_V^{(j)} \oplus W_D^{(j)} \qquad (1)$$

となる。$V^{(j-1)}$は、レベルjの空間の直和になっている。

いま、$f^{(0)}(x,y)$が、空間$V^{(j-1)}$において、次式のように表されているとする。

$$f^{(j-1)}(x,y) = \sum C_{h,k}^{(j-1)} \Phi_{h,k}^{(j-1)}(x,y) \qquad (2)$$

関数$\Phi_{h,k}^{(j)}(x,y)$は、0 あるいは 1 を値とする関数であるので、$C_{h,k}^{(j-1)}$は$f^{(j-1)}(x,y)$の関数値である。関数$f^{(j-1)}(x,y)$を、空間$V^{(j)}$、$W_H^{(j)}$、$W_V^{(j)}$、$W_D^{(j)}$の成分に分解するために、それらの正規直交基底との内積を求める。

$$C_{h,k}^{(j)} = \langle f^{(j-1)}, \Phi_{h,k}^{(j,0)} \rangle = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} + C_{2h+1,2k}^{(j-1)} + C_{2h,2k+1}^{(j-1)} + C_{2h+1,2k+1}^{(j-1)}\right) \times 2^j$$

$$D_{H,h,k}^{(j)} = \langle f^{(j-1)}, \Psi_{H,h,k}^{(j,0)} \rangle = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} - C_{2h+1,2k}^{(j-1)} + C_{2h,2k+1}^{(j-1)} - C_{2h+1,2k+1}^{(j-1)}\right) \times 2^j$$

$$D_{V,h,k}^{(j)} = \langle f^{(j-1)}, \Psi_{V,h,k}^{(j,0)} \rangle = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} + C_{2h+1,2k}^{(j-1)} - C_{2h,2k+1}^{(j-1)} - C_{2h+1,2k+1}^{(j-1)}\right) \times 2^j$$

$$D_{D,h,k}^{(j)} = \langle f^{(j-1)}, \Psi_{D,h,k}^{(j,0)} \rangle = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} - C_{2h+1,2k}^{(j-1)} - C_{2h,2k+1}^{(j-1)} + C_{2h+1,2k+1}^{(j-1)}\right) \times 2^j$$

上記の係数は、正規直交基底に対するものであるが、これらを関数値が 0 または 1 であるものに対する係数で表すと次式を得る。

$$C_{h,k}^{(j)}\Phi_{h,k}^{(j,0)}(x,y) = C_{h,k}^{(j)} \times \frac{1}{2^j}\Phi_{h,k}^{(j)}(x,y)$$

$$= \frac{1}{4}\left(C_{2h,2k}^{(j-1)} + C_{2h+1,2k}^{(j-1)} + C_{2h,2k+1}^{(j-1)} + C_{2h+1,2k+1}^{(j-1)}\right) \times 2^j \times \frac{1}{2^j}\Phi_{h,k}^{(j)}(x,y)$$

$$= C_{h,k}^{(j)} \Phi_{h,k}^{(j)}(x,y)$$

ここで、

$$C_{h,k}^{(j)} = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} + C_{2h+1,2k}^{(j-1)} + C_{2h,2k+1}^{(j-1)} + C_{2h+1,2k+1}^{(j-1)}\right)$$

である。$\Phi_{h,k}^{(j)}(x,y)$の値は 0 または 1 であるので、$C_{h,k}^{(j)}$は関数$f^{(j)}(x,y)$の値を表す（式（２）において、$j-1$をjとおく）。

同様に以下の式が導かれる。

$$D_{H,h,k}^{(j,0)} \Psi_{H,h,k}^{(j,0)}(x,y) = D_{H,h,k}^{(j,0)} \times \frac{1}{2^j} \Psi_{H,h,k}^{(j)}(x,y) = D_{H,h,k}^{(j)} \Psi_{H,h,k}^{(j)}(x,y)$$

$$D_{H,h,k}^{(j)} = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} - C_{2h+1,2k}^{(j-1)} + C_{2h,2k+1}^{(j-1)} - C_{2h+1,2k+1}^{(j-1)}\right)$$

$$D_{V,h,k}^{(j,0)} \Psi_{V,h,k}^{(j,0)}(x,y) = D_{V,h,k}^{(j,0)} \times \frac{1}{2^j} \Psi_{V,h,k}^{(j)}(x,y) = D_{V,h,k}^{(j)} \Psi_{V,h,k}^{(j)}(x,y)$$

$$D_{V,h,k}^{(j)} = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} + C_{2h+1,2k}^{(j-1)} - C_{2h,2k+1}^{(j-1)} - C_{2h+1,2k+1}^{(j-1)}\right)$$

$$D_{D,h,k}^{(j,0)} \Psi_{D,h,k}^{(j,0)}(x,y) = D_{D,h,k}^{(j,0)} \times \frac{1}{2^j} \Psi_{D,h,k}^{(j)}(x,y) = D_{D,h,k}^{(j)} \Psi_{D,h,k}^{(j)}(x,y)$$

$$D_{D,h,k}^{(j)} = \frac{1}{4}\left(C_{2h,2k}^{(j-1)} - C_{2h+1,2k}^{(j-1)} - C_{2h,2k+1}^{(j-1)} + C_{2h+1,2k+1}^{(j-1)}\right)$$

上記の漸化式（下降サンプリング）により、式（１）に基づいて、式（２）における係数$C_{h,k}^{(j-1)}$から空間$V^{(j)}$、$W_H^{(j)}$、$W_V^{(j)}$、$W_D^{(j)}$における係数$C_{h,k}^{(j)}$、$D_{H,h,k}^{(j)}$、$D_{V,h,k}^{(j)}$、$D_{D,h,k}^{(j)}$を求めることができる。

ウェーブレット解析のプログラムをリスト１のように作成した。

リスト１　画像のウェーブレット解析

```
import cv2          # pip install opencv-python
import numpy as np
import matplotlib.pyplot as plt

img1 = cv2.imread('Sample_4.png')     #    画像 512×512 ファイルの読み込み
plt.imshow(img1)
plt.show()
```

```
f_0 = np.sum(img1, axis=2)/3          #   モノクロ化

#
#       ウェーブレット解析結果を辞書型として保存
#
WaveletAnal = {}
WaveletAnal['0'] = {'f0': f_0}

n = 9
N = 2**n

#
#       レベル j の解析結果の図示
#
def disp_map(j):
    Cj = WaveletAnal[f'{j}']['f0']     #   係数 C_h,k
    DHj = WaveletAnal[f'{j}']['DHj']
    DVj = WaveletAnal[f'{j}']['DVj']
    DDj = WaveletAnal[f'{j}']['DDj']
    #
    #    画像を配列 imgs に設定
    #    大きさ 512×512 を４分割して使用
    #
    imgs = np.full((512, 512, 3), 0.5)
    for h in range((2**(n-j))):
        for k in range((2**(n-j))):
            for ii in range((2**j)//2):       #   スケールに合わせて引き延ばし
                for jj in range((2**j)//2):
                    for kk in range(3):     #   赤、緑、青の３チャンネル
                        #
                        #       C_hk の値
                        #
                        imgs[512//2 + ((2**j)//2)*h + ii]¥
                                    [512//2 + ((2**j)//2)*k + jj][kk] ¥
                            = Cj[h][k] / 255
                        #
```

```
                    #     DH, DV, DDは、値０を黒に設定
                    #
                    imgs[512//2 + ((2**j)//2)*h + ii][((2**j)//2)*k + jj][kk] ¥
                        = 0.0
                    imgs[((2**j)//2)*h + ii][(512//2) + ¥
                                            ((2**j)//2)*k + jj][kk] ¥
                        = 0.0
                    imgs[((2**j)//2)*h + ii][((2**j)//2)*k + jj][kk] ¥
                        = 0.0
                if DHj[h][k] > 0:    #   正ならば緑に設定
                    imgs[((2**j)//2)*h + ii][(512//2) + ¥
                                ((2**j)//2)*k + jj][1] ¥
                        = DHj[h][k] / 255
                elif DHj[h][k] < 0:  #   負ならば赤に設定
                    imgs[((2**j)//2)*h + ii][(512//2) +¥
                                ((2**j)//2)*k + jj][0] ¥
                        = -DHj[h][k] /255
                if DVj[h][k] > 0:
                    imgs[512//2 + ((2**j)//2)*h + ii][((2**j)//2)*k + jj][1] ¥
                        = DVj[h][k] / 255
                elif DVj[h][k] < 0:
                    imgs[512//2 + ((2**j)//2)*h + ii][((2**j)//2)*k + jj][0] ¥
                        = -DVj[h][k] /255
                if DDj[h][k] > 0:
                    imgs[((2**j)//2)*h + ii][((2**j)//2)*k + jj][1] ¥
                        = DDj[h][k] / 255
                elif DDj[h][k] < 0:
                    imgs[((2**j)//2)*h + ii][((2**j)//2)*k + jj][0] ¥
                        = -DDj[h][k] /255


plt.imshow(imgs)    #   画像配列 imgs の表示
plt.xticks([])
plt.yticks([])
plt.title(f'j = {j}')
plt.show()
```

```
for j in range(1,n):
    #      レベル j の解析
    print(f'Level {j} started.')
    Cj_1 = WaveletAnal[f'{j-1}']['f0']
    Cj = np.empty((2**(n-j), 2**(n-j)))
    DHj = np.empty((2**(n-j), 2**(n-j)))
    DVj = np.empty((2**(n-j), 2**(n-j)))
    DDj = np.empty((2**(n-j), 2**(n-j)))
    #
    #        下降サンプリング
    #
    for h in range(2**(n-j)):
        for k in range(2**(n-j)):
            Cj[h][k] = (Cj_1[2*h][2*k] + Cj_1[2*h+1][2*k] +
                        Cj_1[2*h][2*k+1] + Cj_1[2*h+1][2*k+1]) / 4
            DHj[h][k] = (Cj_1[2*h][2*k] - Cj_1[2*h+1][2*k] +
                        Cj_1[2*h][2*k+1] - Cj_1[2*h+1][2*k+1]) / 4
            DVj[h][k] = (Cj_1[2*h][2*k] + Cj_1[2*h+1][2*k] -
                        Cj_1[2*h][2*k+1] - Cj_1[2*h+1][2*k+1]) / 4
            DDj[h][k] = (Cj_1[2*h][2*k] - Cj_1[2*h+1][2*k] -
                        Cj_1[2*h][2*k+1] + Cj_1[2*h+1][2*k+1]) / 4
    WaveletAnal[f'{j}'] = {'f0': Cj}
    WaveletAnal[f'{j}']['DHj'] = DHj
    WaveletAnal[f'{j}']['DVj'] = DVj
    WaveletAnal[f'{j}']['DDj'] = DDj

    disp_map(j)     #   解析結果の表示

while True:
    print(f'0 < j < {n}')
    j = int(input('j = '))
    if j <=0 or j >= n:
        break
    disp_map(j)
```

リスト 1 を実行すると、画像ファイル Sample_4.png が読み込まれて表示される（図 1）。画像サイズは 512 ピクセル×512 ピクセルであり、$512 = 2^9$である。

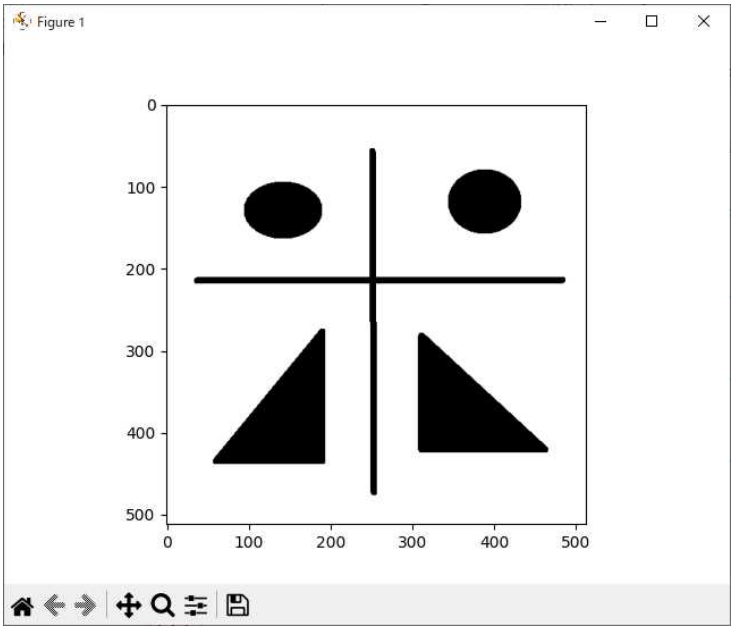

図 1

図 1 のフォームを閉じると、レベル 1 のウェーブレット解析が始まる。レベル 1 の解析が終わると、解析結果が図示される（図 2）。

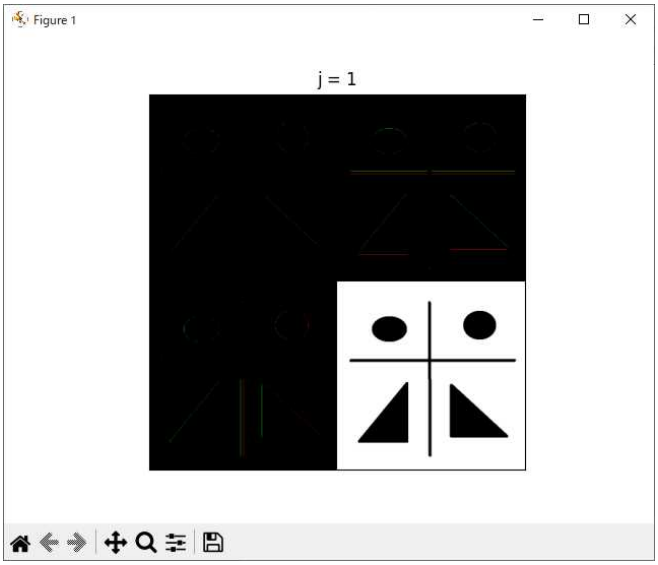

図 2

表示画面の右下に$C_{h,k}^{(j)}$のマップが表示されている。右上は、ウェーブレットの垂直成分、左下はウェーブレットの水平成分、左上が対角成分である。プログラム中では、画像は numpy の ndarray 型で表されているが、表示は OpneCV で行っている。OpenCV での縦座標は ndarray 型の行に、横座標は ndarray 型の列に対応しているが、このことは本稿の最後の附記で確認している。

図２のフォームを閉じると、次のレベル、すなわちレベル２の解析が始まり、解析が終了すると、そお結果が表示される（図３）。

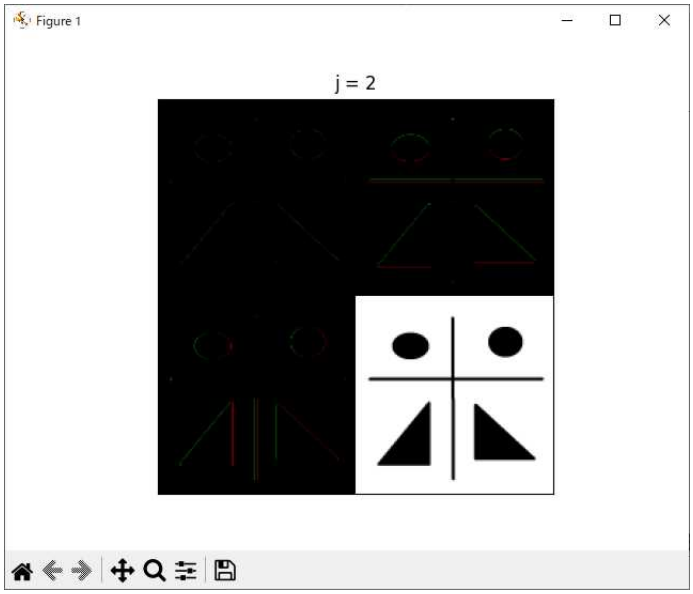

図３

図３のフォームを閉じると、次のレベル３の解析へと進む。この手順を繰り返すと、レベル８まで進む。画像の大きさが 512 ピクセル×512 ピクセルのときは、$512 = 2^9$であるので、レベル８の解析が最終レベルとなる（図４）。

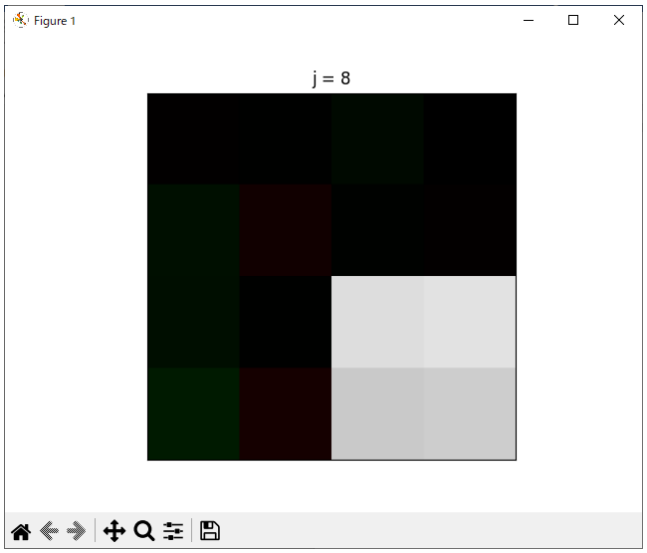

図４

図４のフォームを閉じると、端末画面は以下のようになっている。

```
= RESTART: D:/WaveletAnal/wa_files/WaveAnal.py
Level 1 started.
Level 2 started.
Level 3 started.
Level 4 started.
Level 5 started.
Level 6 started.
Level 7 started.
Level 8 started.
0 < j < 9
j =
```

ｊに解析結果のレベル値ｊを設定すると、そのレベルの結果が表示される。例えば、５を設定すると、下図（図５）のようになる。

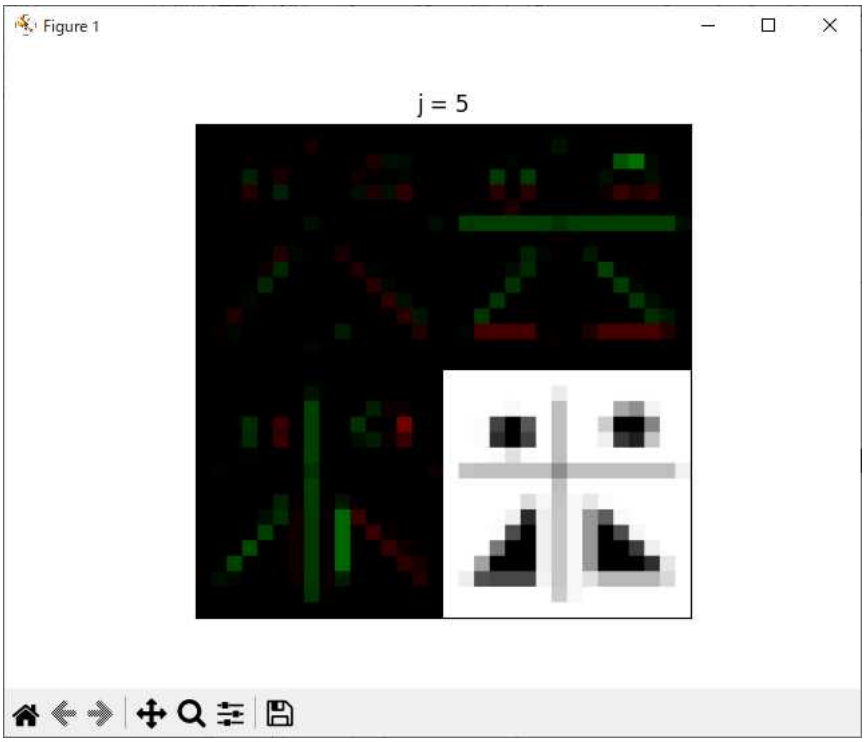

図 5

図 5 のフォームを閉じて、j に範囲外の値、例えば、0 を設定すると、プログラムの実行終了である。

```
0 < j < 9
j = 5
0 < j < 9
j = 0
```

附記　OpneCV の座標軸と numpy の ndarray 型の行・列との関係

画像を表す Numpy の ndarray 型配列の行は、OpenCV の画像において縦軸に対応し、列は OpenCV の画像の横軸に対応している。このことを確認するスクリプトをリスト 2 のように用意した。

リスト２　Numpy の ndarray 型配列と OpneCV の画像との対応

```
import cv2          # pip install opencv-python
import numpy as np
import matplotlib.pyplot as plt

img = np.full((300, 500, 3), 128, dtype = np.uint8)
#
#       Numpy の ndarray 型配列の要素に値を設定
#
for i in range(130, 170):
    for j in range(500):
        img[i][j][1] = 255
#
#      OpenCV の関数で描画
#
cv2.rectangle(img, (225, 0), (275, 300), (0, 0, 255), thickness = -1)
#
#      画像を表す ndarray 型配列の表示
#
plt.imshow(img)
plt.show()
```

リスト２を実行すると図Ａのフォームが表示される。

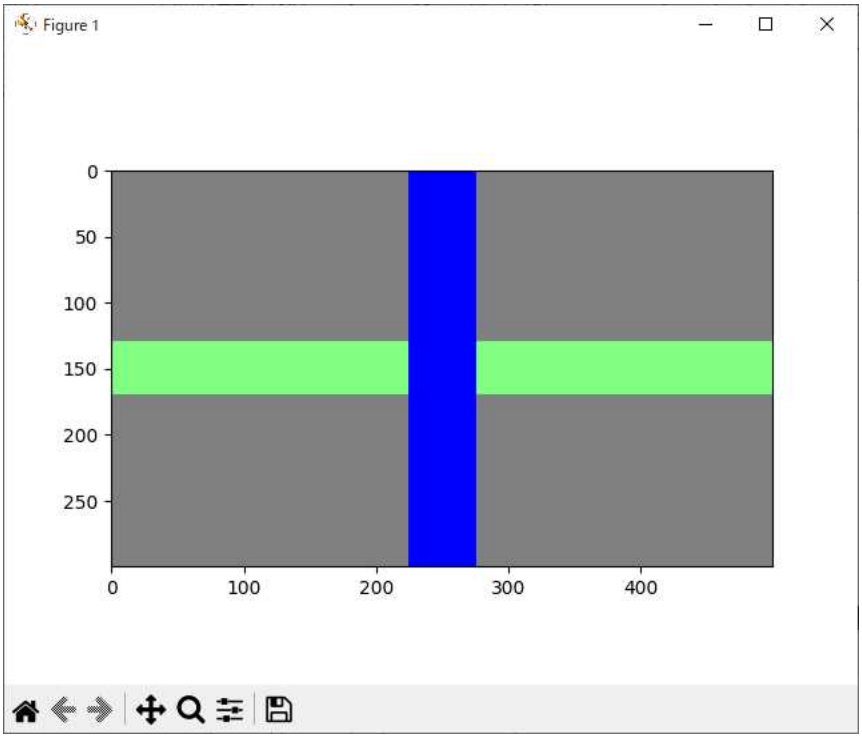

図 A

横の薄緑色の帯が、ndarray 型配列 img の要素に直接値を設定して描いたものである。配列 img の行が OpenCV の画像の縦座標に、配列 img の列が OpenCV の画像の横軸に対応していることが分かる。青色の縦の帯は、OpenCV の関数 rectangle で描いたものである。第 1 座標が横座標に、第 2 座標が縦座標に対応していることが分かる。

引用文献

金谷健一（2003）これなら分かる応用数学教室．共立出版
Nason, G. P. (2008) *Wavelet methods in statistics with R*. Springer