「画素差分値≒近傍画素との距離」となる2D三角波を作る

今回の実験で使った「背景画像」は、図5のような「2次元（2D）の三角波」です。

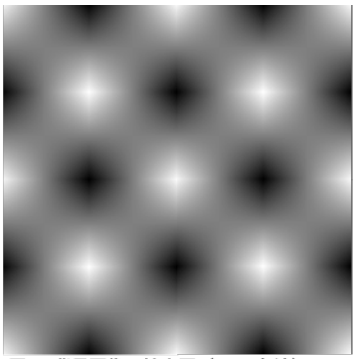

図5　背景画像の拡大図（2D三角波）

2D三角波を使った理由は、『三角波を使うと「画素値差をとる」で「大雑把に変位量を見積もる」ことができる』からです。図6は、1次元で単純化すると、**『三角波は「Nピクセル離れた場所の画素値」が「自画素値」とNに比例した量だけの差を持つ』**ことを図示したものです。こうした特性があるため、画像差分をとるだけで、簡単にシュリーレン計測ができる！というわけです。

正確に言えば、2次元の場合、本来は2D三角波ではなくて「交互に反転する円錐」を繰り返す必要があります。とはいえ、2D三角波画像との違いは少ないことや、将来的に「横方向・縦方向の画素値変化を、それぞれに個別色を使う」ことを考えて、たとえば、

・横方向は緑色の三角波を使う
・縦方向は赤色の三角波を使う

といった拡張を行うことを考えて、今回は2D三角波画像を使っています。

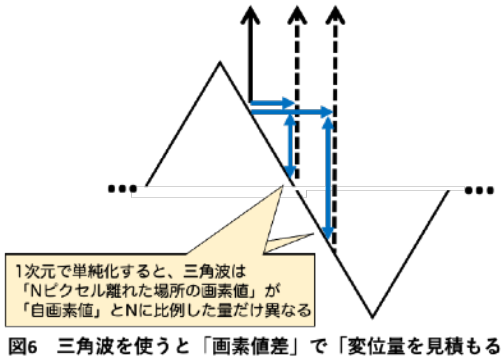

図6　三角波を使うと「画素値差」で「変位量を見積もる」ことができる

```python
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
import cv2

# 画像縦横サイズ
wh = 256
# wh x wh x 1チャンネル、8bit画像
x = np.arange(wh)
y = np.arange(wh)

# 2D三角波のピクセル周期
pitch_in_pix = 20
n = wh / pitch_in_pix

# 2D三角波の作成関数
def saw2d(x,y):
    return 127*signal.sawtooth(2*np.pi*n*x/wh,0.5)\
              *signal.sawtooth(2*np.pi*n*y/wh,0.5)+127
# 背景画像を作る
img = np.zeros((wh, wh, 1), np.uint8)
for y in range(wh):
    for x in range(wh):
        img[y][x] = saw2d(x,y)
# 画像確認
plt.imshow(img, cmap = "gray")
# 画像保存
cv2.imwrite('background.png', img)
```

図7　2D三角波画像を作り保存するPythonコード

今回使った背景画像、2D三角波画像を生成・ファイル保存するPythonコードが図7です。numpyを使った数行のコードです。

実現が難しく思えることも、「少しの工夫」で「とても簡単」になる

目に見えない気流の可視化も、スマホとPython数行で実現できます。……そんなことができるのは、Pythonや画像処理ライブラリーOpenCVの便利な機能のおかげでもありますが、背景画像の選択など「少しの工夫」をしていることも、大きな要因です。

実現が難しく思えることも、少し見方を少し変えたり、少し工夫をしたりすると、とても簡単に実現することができたりします。そのためには、色んな「眺め方」や「道具」を知っておくことが役立ちます。何でも出てくるドラえもんポケットのように、あなたの「引き出し」を広くすることに役立つことができれば良いな、と思います。

本記事コード・補足説明：https://github.com/hirax/SD202210