澤田 英宏

画像処理例がカラーで見られます（https://interface.cqpub.co.jp/2021-1-1-10/）

10-1 テンプレート・マッチング

CPU版① CPU版② GPU版

プログラム名：①MatchTemplate.py，②MultiMatchTemplate.py
（GPU版はMatchTemplate_gpu.py）
CPU版 https://interface.cqpub.co.jp/10-1match-template-py/，/10-1multimatchtemplate-py/
GPU版 /10-1matchtemplate_gpu-py/

● 概要

テンプレート・マッチングは，関数がシンプルに使用できるにもかかわらず，検出精度は高いです．提供するサンプル・プログラムでは，

1，画面全体からマッチする1つを探す方法
2，画面全体からマッチするもの全てを探す方法

の2つを解説しています．

判定には，どちらも同じ関数を使っていますが，検出結果として，

- リスト1は最も高い値の座標を見つける
- MultiMatchTemplate.pyはしきい値以上の座標を探す

という違いとなります．

● 前処理

この機能では前処理としてグレー・スケール化を使っています．詳しくは「2-1　グレー・スケール化」を参照してください．

● CPU版その1：画面全体からマッチする部分を1つ探すプログラム…MatchTemplate.py

リスト1は画像全体の中から1つだけ合致する部分を探し出す方法です．

▶リスト1：008行…マッチする部分を探す

実際にテンプレート・マッチングを実行します．

```
result = cv2.matchTemplate(image=img, templ=temple, method=cv2.TM_CCOEFF)
```

第1引数imageは入力画像です．
第2引数templは参照画像です（図1）．
第3引数methodは検出アルゴリズムを指定します（OpenCVで定義された定数を指定）．検出アルゴリズムには次の6種類が設定されています．

cv2.TM_SQDIFF，cv2.TM_SQDIFF_NORMED
2乗差分法と呼ばれ，テンプレート画像と入力画像の画素の差分を2乗したものの総和を計算．2番目の関数でデータ正規化した場合は，値が0に近いほど似ていると判断できる．

cv2.TM_CCORR，cv2.TM_CCORR_NORMED
相互相関法と呼ばれ，テンプレート画像と入力画像の画素同士の内積の総和を計算．2番目の関数でデータ正規化した場合は，値が1に近いほど似ていると判断できる．

cv2.TM_CCOEFF，cv2.TM_CCOEFF_NORMED
相関係数法と呼ばれ，テンプレート画像の平均値と入力画像の平均値画素同士の内積の総和を計算．2番目の関数でデータ正規化した場合は，値が1に近いほど似ていて，－1に近いほど似ていないと判断できる．

リスト1ではTM_CCOEFFを使って検出するようになっていますが，cv2.TM_SQDIFF，cv2.TM_SQDIFF_NORMEDを使用する場合は，判定結果が逆のため012行でmaxLocの代わりにminLocを使用しないと間違った場所を示します．これらのアルゴリズムは，検出したい場面ごとに固定された手法があるわけではなく，期待する結果を示さない場合もあります．

▶リスト1：011行…最大値と最小値を見つける

008行で取得したデータをもとに，最大値の座標を見つけます．

```
minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(result)
```

第1引数resultはマッチング・データです．
第1戻り値minValは最小値です．
第2戻り値maxValは最大値です．
第3戻り値minLocは最も類似する座標です．
第4戻り値maxLocは最も類似しない座標です．

リスト1　テンプレート・マッチング（1つ検出）プログラムCPU版（MatchTemplate.py）

```
000: import cv2
001: import numpy as np
002:
003: def __main():
004:     img = cv2.imread('../IMG_1630_2.jpg')
005:     temple = cv2.imread('../IMG_1630_2_T.jpg')
006:     h, w, _ = temple.shape
007:
008:     result = cv2.matchTemplate(image=img,
                templ=temple, method=cv2.TM_CCOEFF)
009:
010:     # 配列内のグローバルな最小値と最大値を見つける
           minLoc=最も類似する座標 maxLoc=最も類似しない座標
011:     minVal, maxVal, minLoc, maxLoc =
                                cv2.minMaxLoc(result)
012:     x, y = maxLoc
013:     color = np.array([0., 255., 255.]) # BGR表記
014:     cv2.rectangle(img=img, pt1=(x, y),
       pt2=(x + w, y + h), color=color, thickness=3)
015:
016:     cv2.imshow('Final result', img)
017:     cv2.imshow('Template', temple)
018:     cv2.waitKey(0)
019:     cv2.destroyAllWindows()
020:
021:     return 0
省略
```