ダウンロード・データあります



ご購入はこちら

# 差分やエッジの検出

土井 伸洋

## 4-1，ラベリング

ラベリングをひと言で言うと，領域ごとの塗り分けです．2値化された画像に対して，連続した白い部分に同じIDを割り振る処理です．同じIDごとの座標や面積，輝度などの特徴量を求めることで，抽出処理や分類処理などに利用します．どちらかといえば単体で使われることは少なく，2値化/モルフォロジー変換と一緒に使われます．

リスト1　画像のラベリング

```
# 2値化
# * 大まかなコイン領域を得る
gray_image = cv2.cvtColor(src_image,
                                    cv2.COLOR_RGB2GRAY)
_, binary_image = cv2.threshold(gray_image,
thresh=160, maxval=255, type=cv2.THRESH_BINARY_INV)

# モフォロジー変換
# * ノイズを除去．画像の状況からクロージング処理のみで対応した．
kernel = np.ones((7, 7), dtype=np.uint8)
closing_image = cv2.morphologyEx(binary_image,
                    op=cv2.MORPH_CLOSE, kernel=kernel)

# ラベリング
n_labels, labels = cv2.connectedComponents(
                                            closing_image)
# ラベルイメージの描画
label_image = np.zeros_like(src_image)
for label_id in range(1, n_labels):
  label_pixel_indexes = np.where(
                                    labels == label_id)
  label_image[label_pixel_indexes] =
                np.random.choice(range(256), size=3)
```

### ● プログラムと実行結果

ここでは実践的な例として，硬貨の抽出を行います．プログラムを**リスト1**に示します．

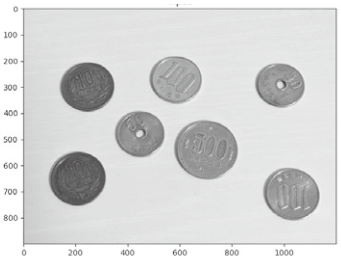
ここで，`cv2.connectedComponents()`関数の返り値は2つです．

- `n_labels`：ラベル数．背景も1つと数える
- `labels`：ラベルIDが振られた配列（入力画像と同サイズ）

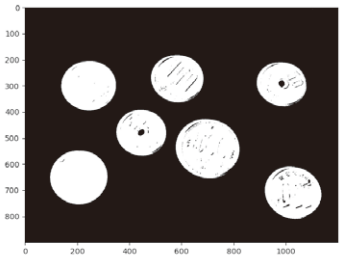
結果を視覚化したものが**図1**です．それぞれの硬貨の領域が個別色できれいに塗りつぶされています．

なお，`connectedComponents()`関数以外にも，

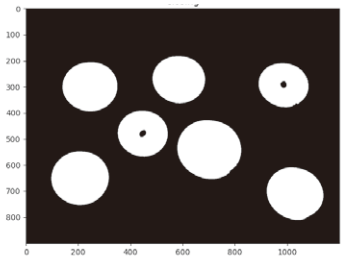
- `connectedComponentWithAlgorithm()`関数



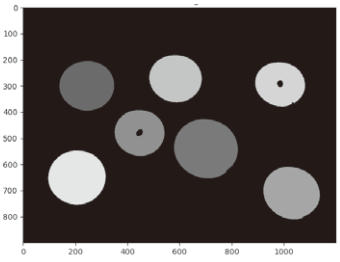
（a）入力画像



（b）2値化



（c）クロージング



（d）ラベリング

**図1　リスト1（画像のラベリング）の実行結果**

非常に良好な結果が得られた．入力画像が単純であれば，深層学習によるセグメンテーション処理を行わずとも，基本的画像処理技術だけで十分結果を出せる場合もある