2020年度　前期 画像処理および演習　第10週

テンプレートマッチング

情報科学科
末永、森本、澤野、塚田

今回の内容

- 講義
  - ◆ テンプレートマッチングの理解
    原理、方法、応用例
- 演習
  - ◆ 類似度（不一致度）　【手計算】
  - ◆ ウォーリーを探せ　　【プログラミング】
- 課題

テンプレート（大辞泉より）

1　型板。

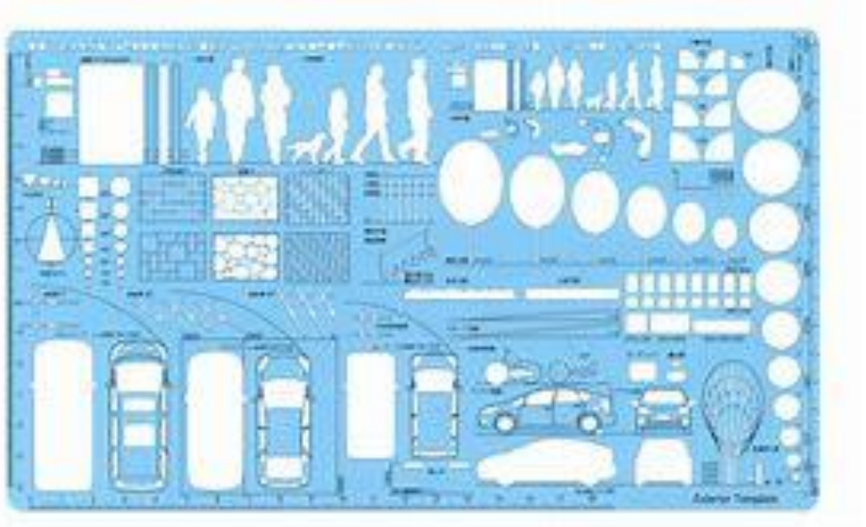
2 四角形や円などがくりぬいてある定規板。内側の縁に沿って図形を描くもの。

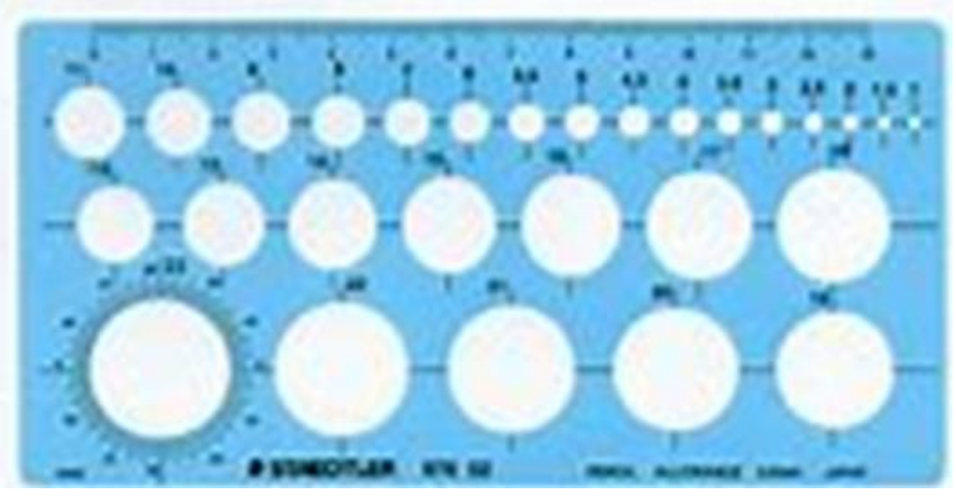
3 歯のかみ合わせを矯正する器具の一種。

4 パソコンのキーボード上に置く、キーの機能を表示したシート。

5 パソコンで、表計算やデータベース用ソフトのサンプル集。形式を整えてあり、すぐに利用できる。フォーム。

6 ⇒定型文

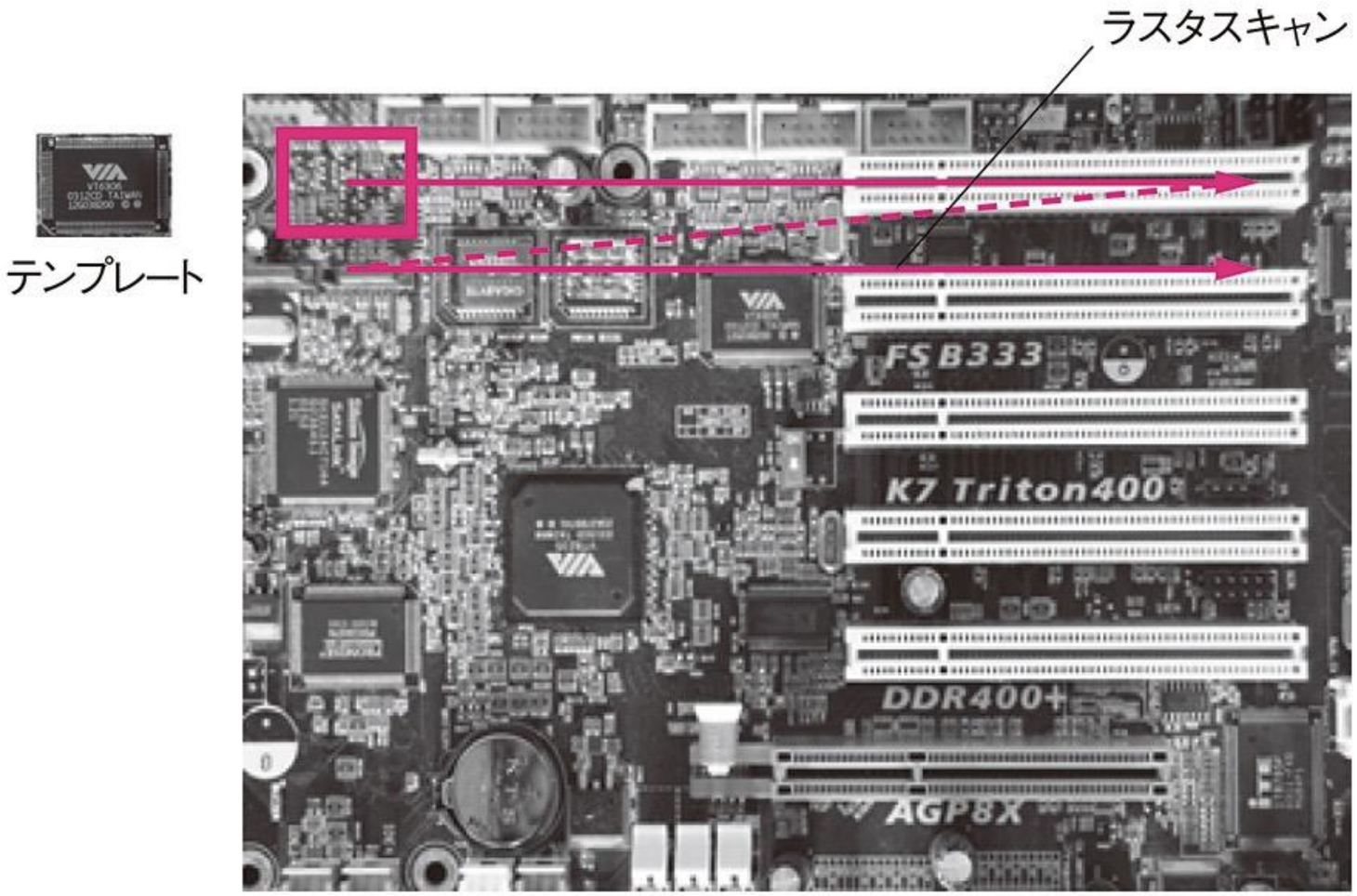




テンプレートマッチングとは

- 入力画像の中からテンプレートに近い領域を検出

■図11.1——テンプレートマッチングにおける
テンプレートの移動の例

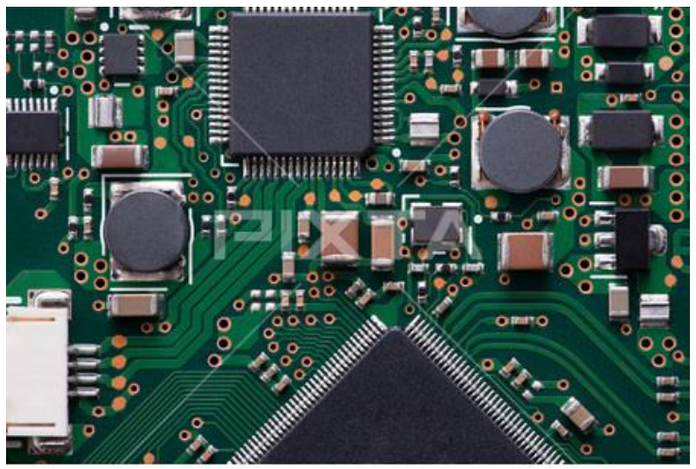


「ディジタル画像処理　改訂新版」2015年 ／ 公益財団法人 画像情報教育振興協会(CG-ARTS協会)

どんなところで使用されるか

- 組み立て検査（基板搭載部品検査）　正解との比較
- 錠剤の調査（使用している錠剤の探索）　色形大きさ
- ウォーリーを探せ　　探したい姿形が既知の場合

検出し（見つけ）たい対象が既知（画像で提示）であるもの　　形、色、明るさ、大きさ、傾き



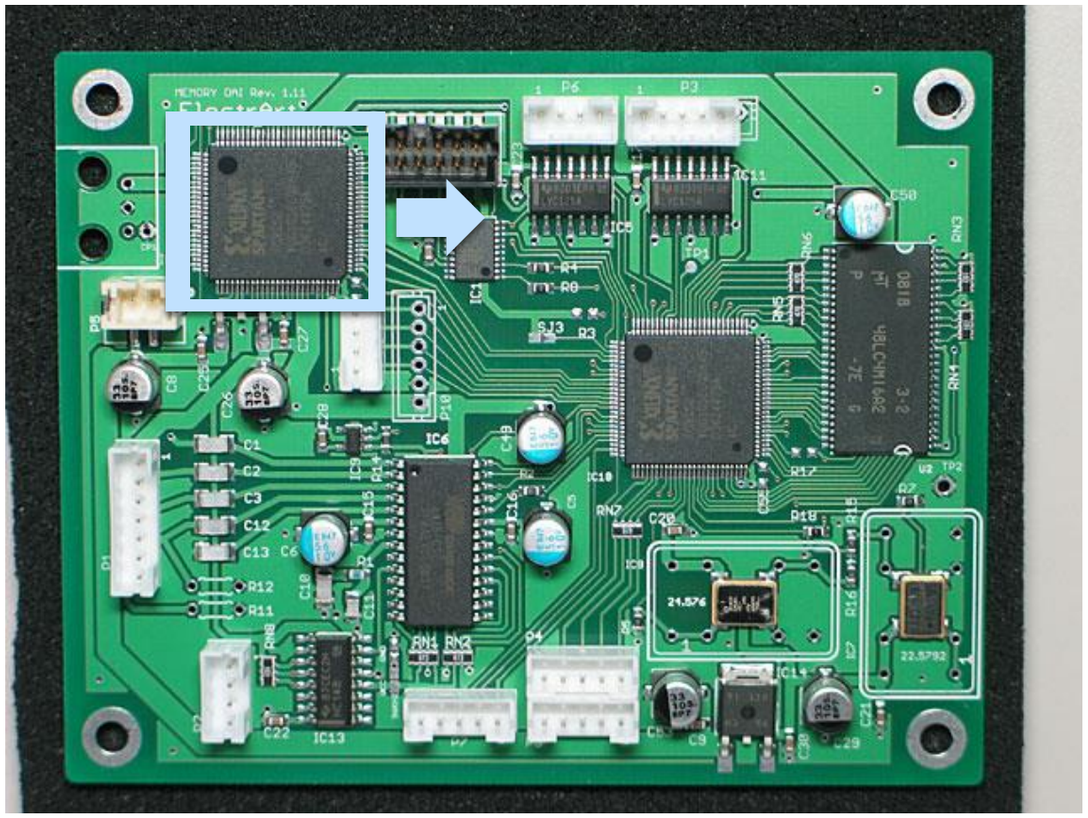
得意：機械部品

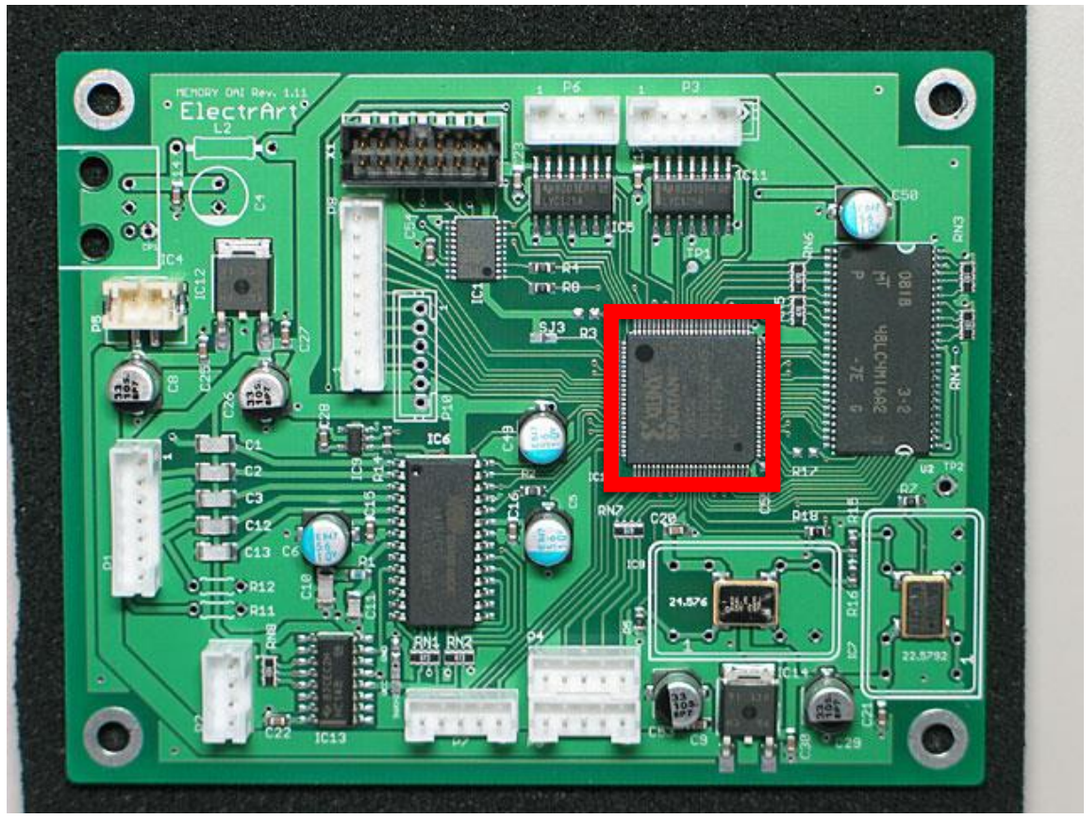
苦手：人の顔

実装部品検査の例

- 搭載部品の有無/極性の検査

テンプレートを走査して部品を探索





回転や拡大/縮小に対しては脆弱

顔検出への応用例

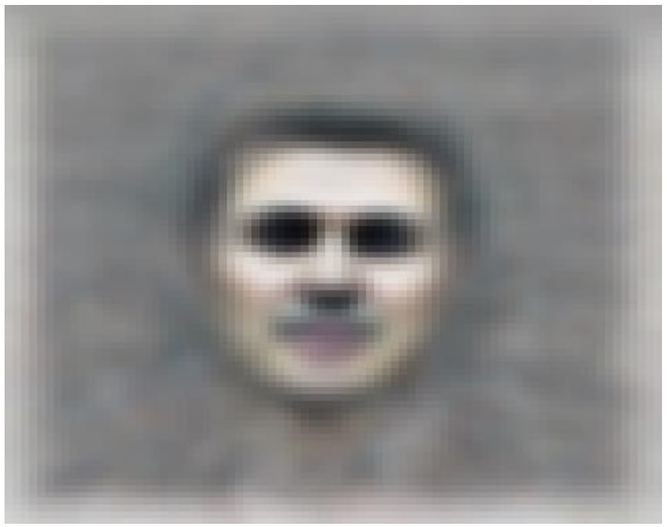
- https://codezine.jp/article/detail/86


原画像

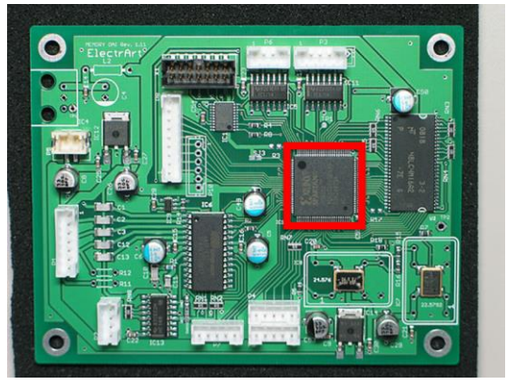

濃淡画像


二値画像


テンプレート


Cf：googleの猫認識（2012）

類似度（不一致度）



- テンプレートと画像がどれだけ近いかの指標

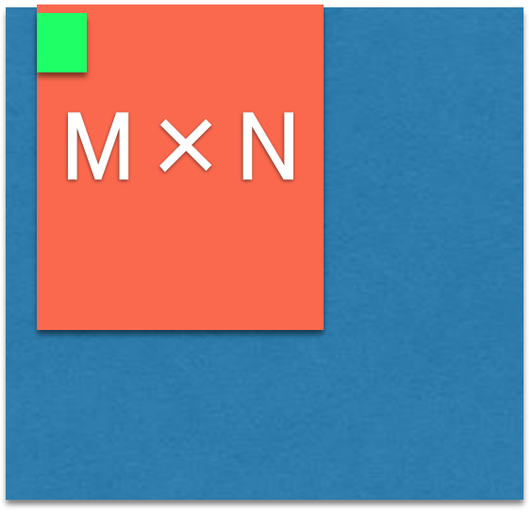
$$R_{SSD}(i,j) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\left(I(i+u,j+v) - T(u,v)\right)^2$$

差の2乗の和

$$R_{SAD}(i,j) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\left|I(i+u,j+v) - T(u,v)\right|$$

差の絶対値の和

- テンプレートのサイズ: $M \times N$
- テンプレート内の位置: (u, v)



- テンプレートと画像が一致した時に値は0になる

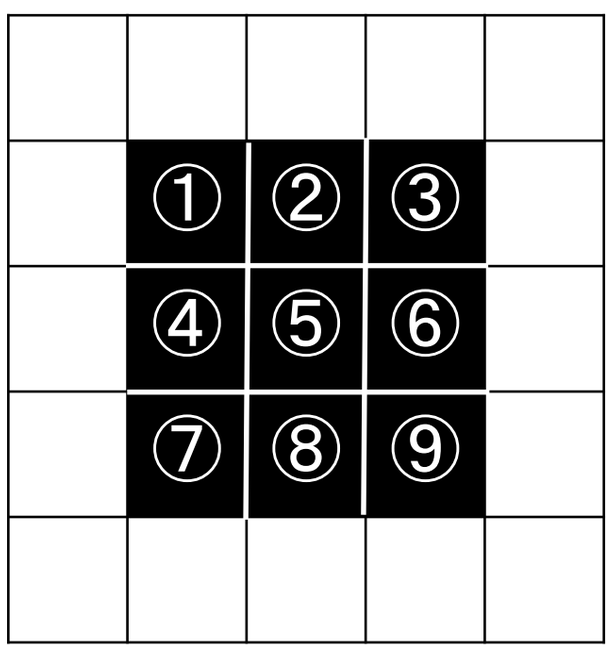
SSD：Sum of Squared Difference/SAD：Sum of Absolute Difference

類似度演算

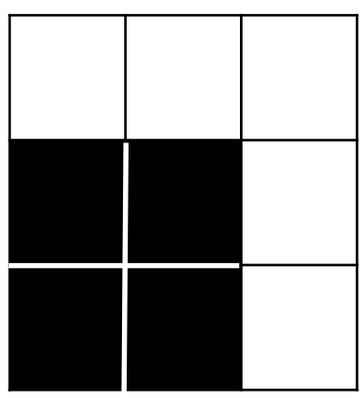
演習

◆ 白:0、黒:1 として、①～⑨の位置における
テンプレートとの類似度（SAD差の絶対値の和）
を求めよ。

抽出対象画像
（5×5）



テンプレート
（3×3）



（答え）

①＿＿＿

②＿＿＿

③＿＿＿

④＿＿＿

⑤＿＿＿

⑥＿＿＿

⑦＿＿＿

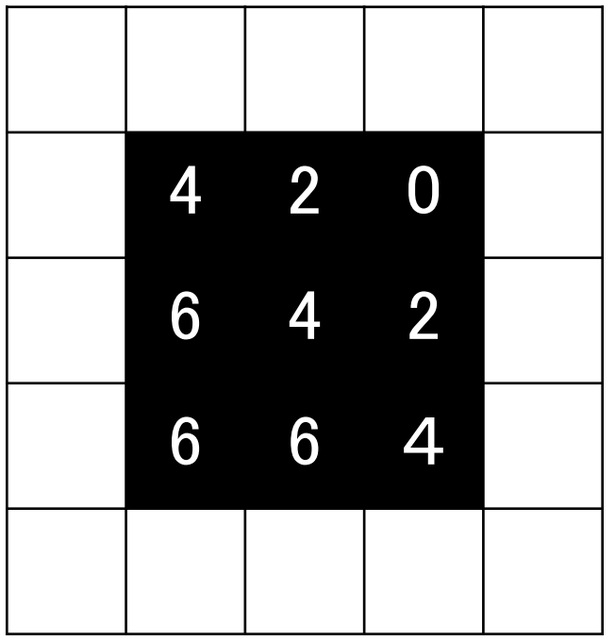
⑧＿＿＿

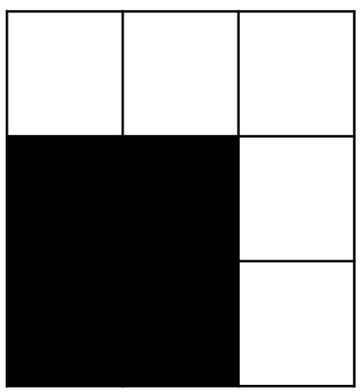
⑨＿＿＿

演習　類似度（不一致度）演算

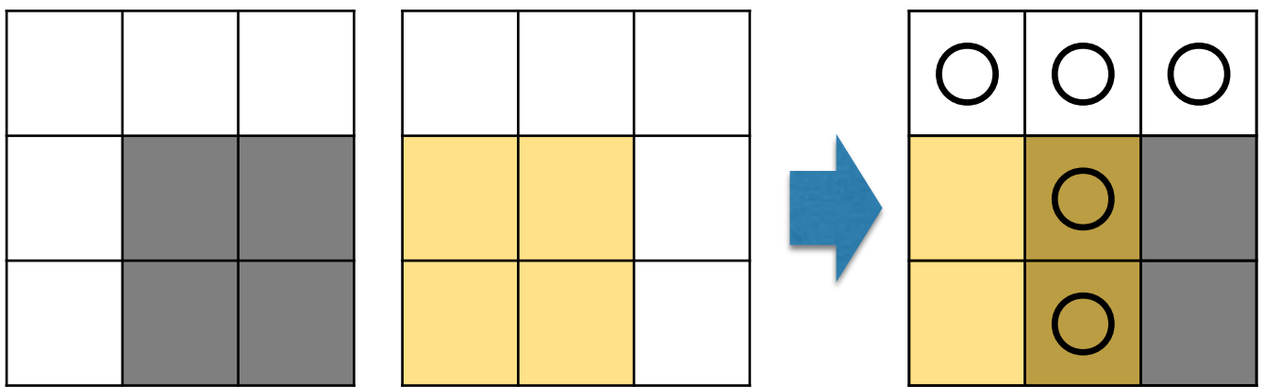
◆ 白:0、黒:1 として、テンプレートとの類似度を
①～⑨の位置において求めよ。

抽出対象画像





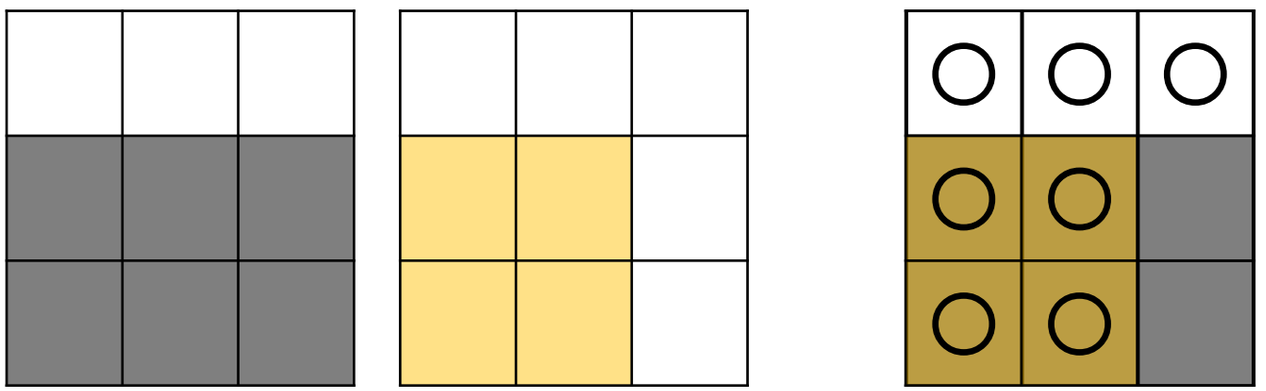
①



テンプレート

4

②

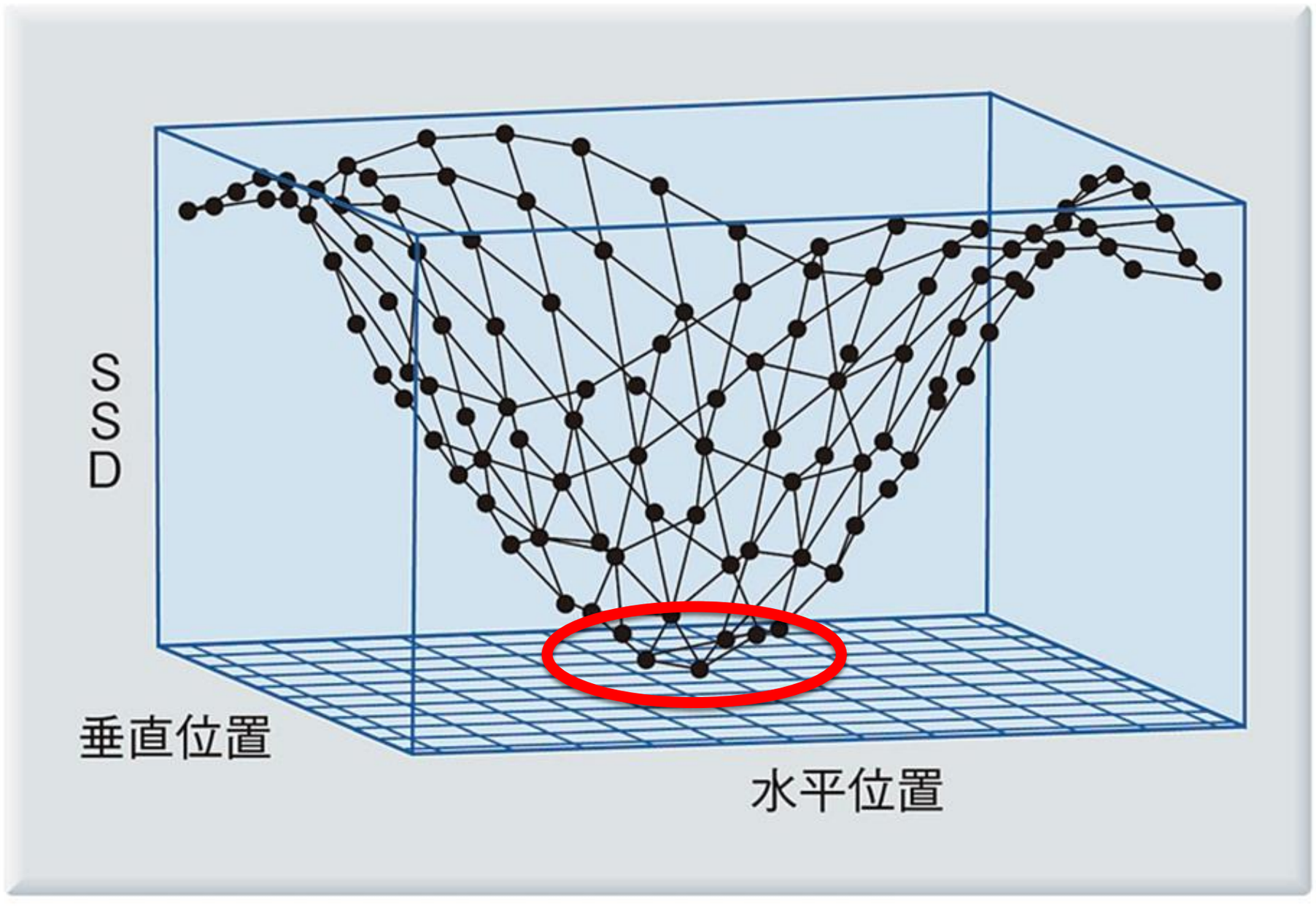


2

③ 0　④ 6　⑤ 4　⑥ 2

⑦ 6　⑧ 6　⑨ 4

類似度（不一致度）の値のイメージ



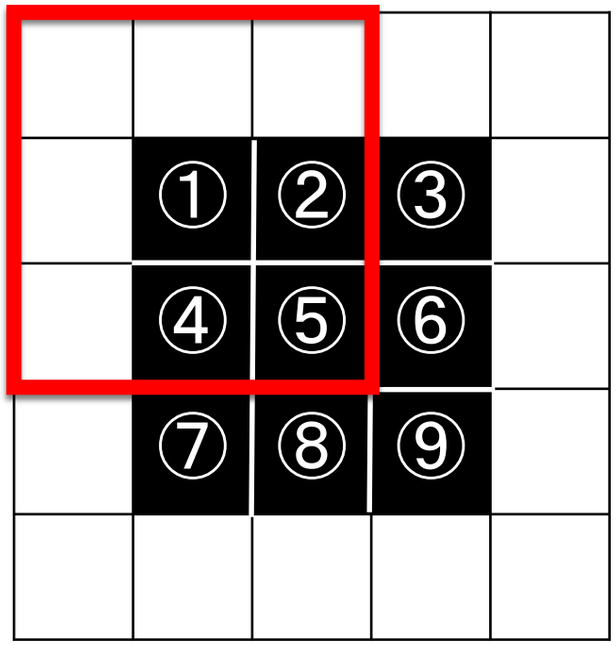
「ディジタル画像処理　改訂新版」2015年／公益財団法人 画像情報教育振興協会（CG-ARTS協会）

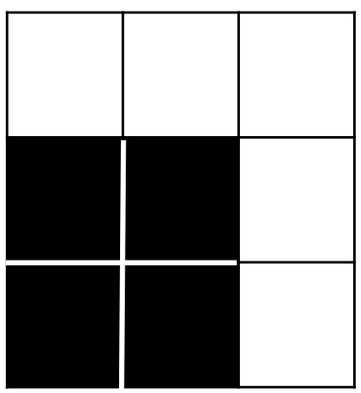
テンプレートの位置表現

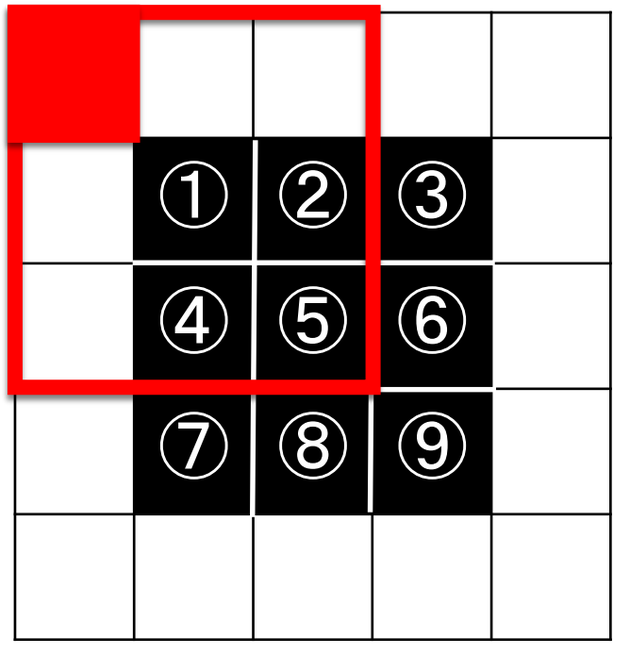
演習では①～⑨で表現した ⇒ テンプレート左上座標

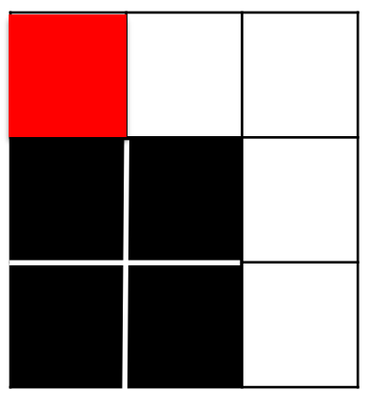
抽出対象画像　テンプレート

（5×5）　（3×3）









対象を検出する画像処理技術

① 照合処理　テンプレートマッチング

② 特徴記述/抽出法

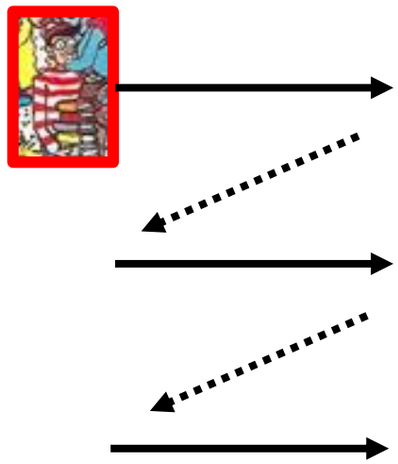
③ 学習法（ニューラルネットワーク）

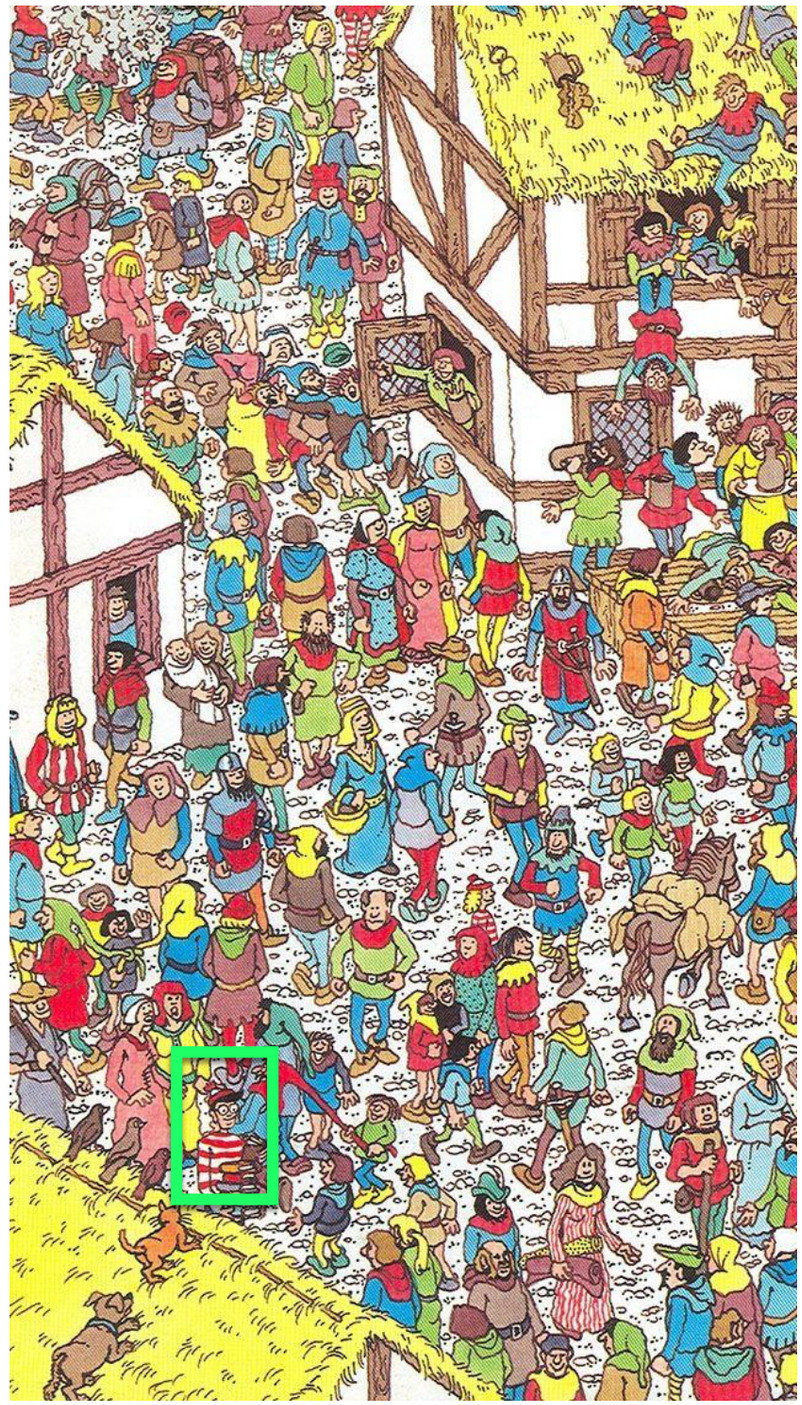
ウォーリーを探せ

① 照合処理 テンプレートマッチング

◆ 見つけたいモノの見え方
（形・大きさ）が既知の場合

見つけたいモノ
＝テンプレート





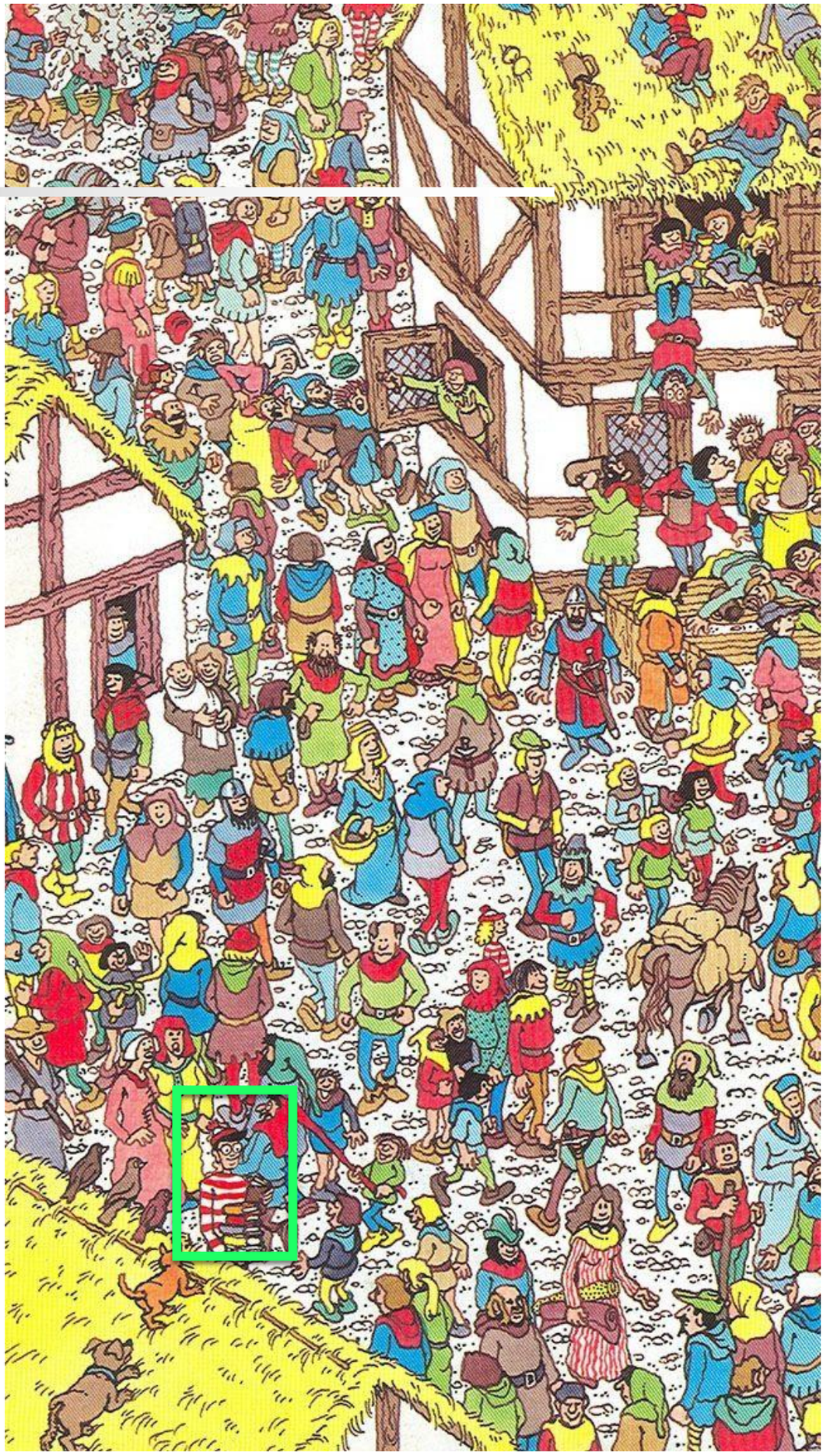
テンプレートを
画像全体に走査して探査

② 特徴記述/抽出法

発見したいモノの
特徴を記述して抽出する

◆ 赤白の縞シャツ　◆ 青色ズボン
◆ 赤白の帽子　　◆ ステッキ　、、、

そのままじゃない！



② 特徴記述/抽出法

ウォーリーはどこ?

赤白の横縞シャツ
青色の長ズボン
ステッキ．．．

こんな人がいます！
これは違うよね。

③ 学習法（N.N.）

ウォーリーを学習

学習によって特徴を抽出

演習　ウォーリーを探せ！

前準備

- サンプルファイルのダウンロード
  - main.cpp

サンプル画像のダウンロード

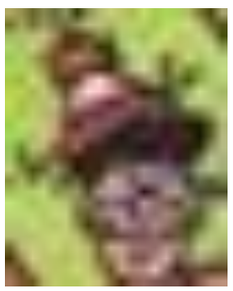
- bg.jpg（入力画像）
- face.jpg（テンプレート）

- 変数概要
  - target_img: 入力画像　　bg.jpg
  - template_img: テンプレート　　face.jpg
  - compare_img: 類似度マップ

➡サイズ(縦横共に): 入力画像−テンプレート+1

➡1チャンネル

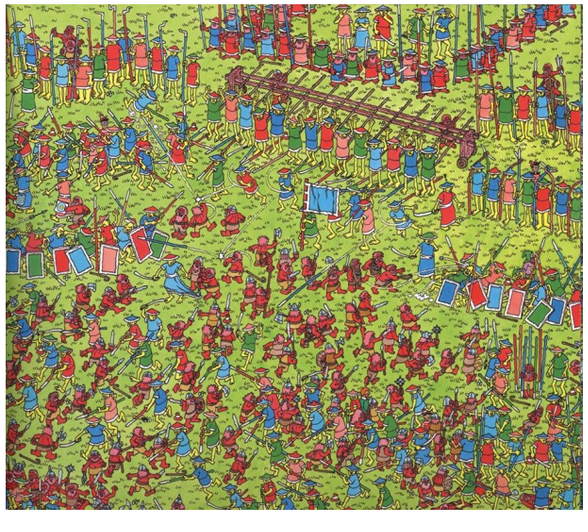
bg.jpg



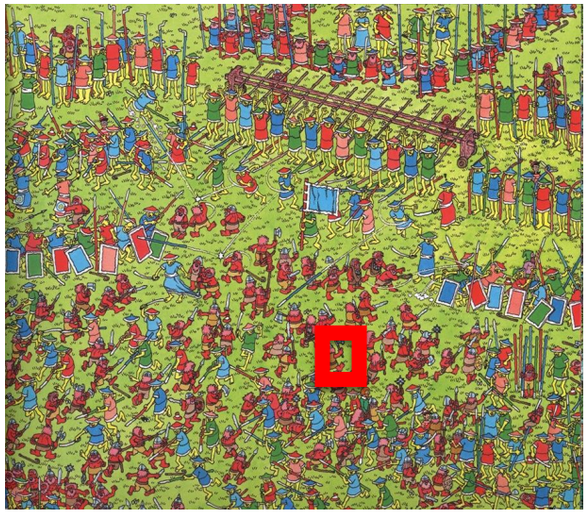
face.jpg

処理の流れ

1. 入力画像とテンプレートの読み込み
2. テンプレートマッチング（類似度マップ作成）
3. 目的画像探索（類似度マップ参照）
4. 目的画像の位置に長方形の描画
5. 表示



類似度
マップ



```cpp
#include <iostream>
#include <opencv2/opencv.hpp>

//---------------------------
//マクロ

//全体の画像（探索対象画像）のファイル名
#define TARGET_IMG_FILE "bg.jpg"

//テンプレート画像
#define TEMPLATE_IMG_FILE "face.jpg"
//結果画像
#define RESULT_IMG_FILE "result.jpg"


//ウィンドウの名前
//探索対象
#define TARGET_IMG_WINDOW "target"
//テンプレート
#define TEMPLATE_IMG_WINDOW "template"


//---------------------------

int main(int argc, const char * argv[]) {

    double min_val, max_val;    //最小値，最大値
    cv::Point min_loc, max_loc; //最小値の位置，最大値の位置
    //全体画像，テンプレート画像，類似度マップ
    cv::Mat target_img, template_img, compare_img;

    //カラーで取得
    target_img = cv::imread(TARGET_IMG_FILE); //全体画像
    template_img = cv::imread(TEMPLATE_IMG_FILE); //テンプレート画像
    if (target_img.empty() || template_img.empty()) { //入力失敗の場合
        fprintf(stderr, "File is not opened.¥n");
        return (-1);
    }

    //テンプレートマッチング前に表示
    cv::imshow(TARGET_IMG_WINDOW, target_img);
    cv::waitKey();

    //テンプレートマッチング

    //類似度マップから次の値の取得．最小値，最大値，最小値の位置，最大値の位置

    //長方形の表示

    cv::imshow(TARGET_IMG_WINDOW, target_img);
    cv::waitKey();

    //画像の保存
    cv::imwrite(RESULT_IMG_FILE, target_img);


    return 0;
}
```

処理の流れ

1. 入力画像とテンプレートの読み込み
2. テンプレートマッチング（類似度マップ作成）
3. 目的画像探索（類似度マップ参照）
4. 目的画像の位置に長方形の描画
5. 表示

テンプレートマッチング

- OpenCVで関数が用意されている

cv::matchTemplate (入力画像, テンプレート, 類似度マップ,
cv::TM_SQDIFF_NORMED);

- cv::TM_SQDIFF: 類似度計算のフラグ

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

- cv::TM_SQDIFF_NORMED: 正規化. 値域[0:1]

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- cv::TM_CCORR: テンプレートと画像の積の和

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

OpenCV 4　https://docs.opencv.org/4.3.0/df/dfb/group_imgproc_object.html#ga3a7850640f1fe1f58fe91a2d7583695d
OpenCV 旧バージョン　http://opencv.jp/opencv-2svn/cpp/object_detection.html

類似度の調査

- 類似度の最大値最小値の調査
  - どこが似ているか，似ていないかの調査（位置と値）
  - 宣言

```
double min_val, max_val;    //最小値，最大値
cv::Point min_loc, max_loc; //最小値の位置，最大値の位置
```

  - 計算

```
cv::minMaxLoc (類似度マップ, &min_val, &max_val,
                              &min_loc, &max_loc);
```

長方形の描画

- 最も似ている部分に長方形の描画
- 長方形の大きさはテンプレートと同じ
- 2頂点必要

長方形の描画関数

- 類似度が最小の位置に長方形の描画

cv::rectangle (入力画像, 矩形の頂点, 対角の頂点, 色値, 太さ);

- 入力画像: target_img
- 矩形の頂点 (cv::Point型): min_loc
- 対角の頂点 (cv::Point型)(テンプレートのサイズで対角の決定):
  cv::Point (min_loc.x + template_img.cols,
  min_loc.y + template_img.rows)
- 色値: CV_RGB (255, 0, 0)
- 太さ: 3

閾値以下の位置を描画する場合（複数箇所抽出）

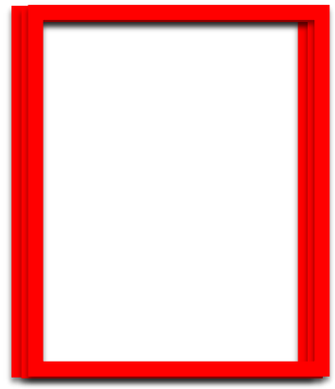
```
float s; //類似度マップはfloat型
for (int y=0; y<compare_img.rows; y++) {
    for (int x=0; x<compare_img.cols; x++) {
        s = compare_img.at<float>(y, x); //float型で取得
        if (s < 0.1) { //閾値以下
            //長方形の表示
            cv::rectangle(target_img, cv::Point(x, y),
                cv::Point(x + template_img.cols,
                          y + template_img.rows),
                          CV_RGB(255, 0, 0), 3);

            fprintf(stderr, "%d, %d¥n", x, y);
        }
    }
}
```

出力結果

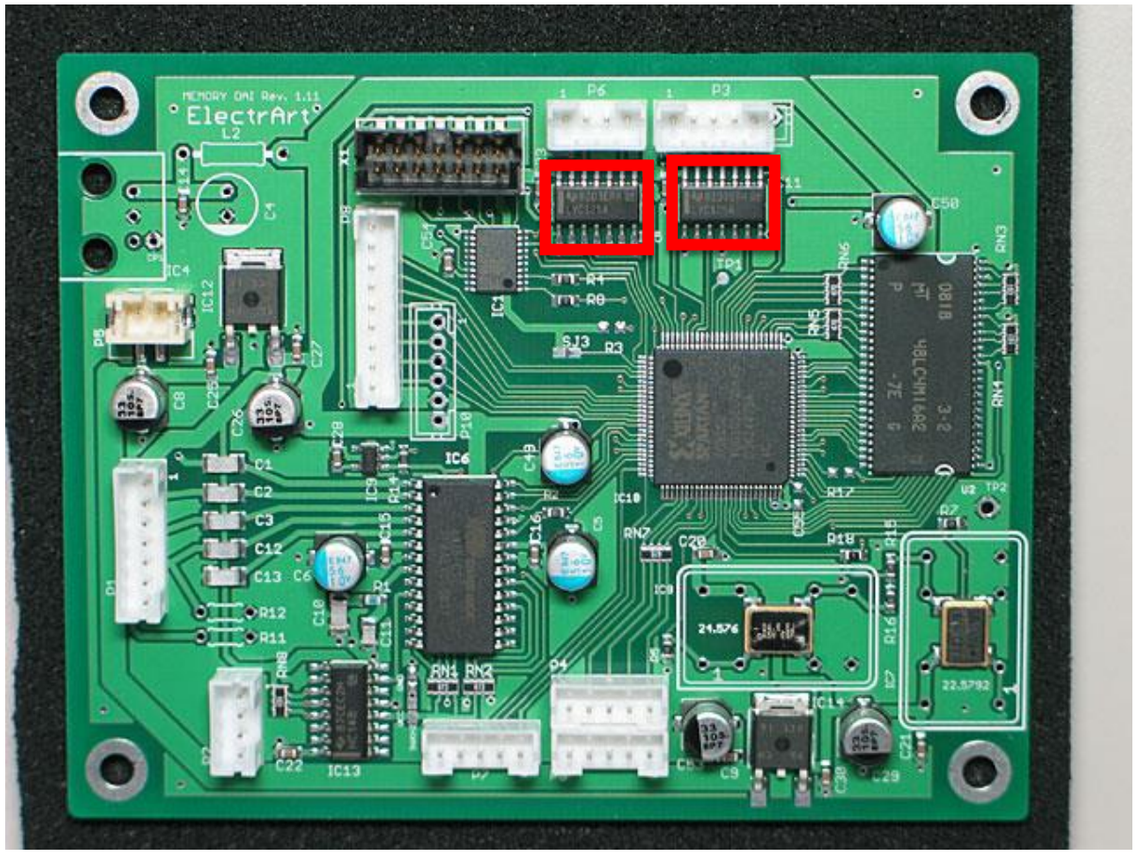
- 近い位置で検出される

141, 168
142, 168
143, 168



テンプレートマッチングの応用

- 検出したい対象が複数ある場合



複数の物体の検出

- テンプレートマッチングで、cv::minMaxLoc()を用いると1物体のみ検出
- 類似度マップを閾値処理で該当箇所の限定
  - 類似度マップ（正規化済み）で閾値処理

```
//閾値処理
cv::threshold(compare_img, compare_img, 0.1, 1.0,
                                        cv::THRESH_BINARY);
```

  - 閾値以上(0.1)を1.0，それ以外は0