画像処理及び演習

第7回 二値画像処理（1/2）

二値画像処理

- 二値化画像とは
- 二値化処理　P-タイル法/判別分析法
- 連結性
- 膨張・収縮処理

- ラベリング、輪郭追跡、・・・・・

二値化　　（教科書　p.59）

ノート

- 白（255）もしくは黒（0）の画像に変換する事
- グレースケール画像を二値化する場合、
  ある値（輝度値）を基準に画素値を変換

  基準の画素値：しきい値、閾値、Threshold

- 画像から位置・形状の情報に変換
  ⇒ 幾何学的に扱うことが可能

二値画像はどんなとき使うか

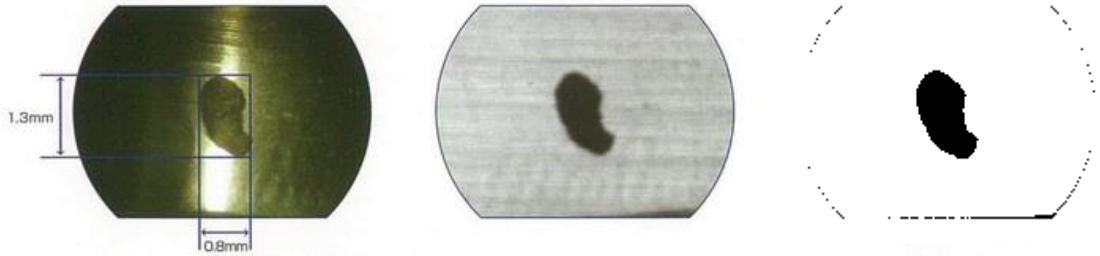
◆ 画像の明瞭化 （見た目）

入力画像

二値画像

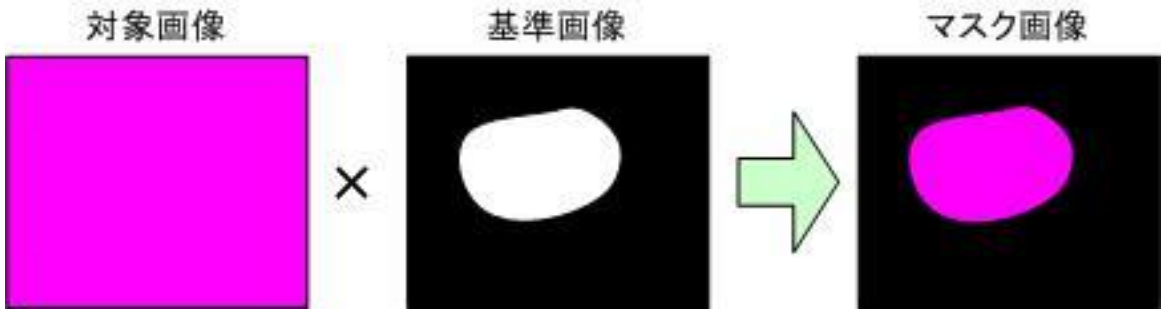
応用例(形状)

- 部品の欠陥を見つける処理



- 領域分割、抽出

- マスク処理



二値化処理：固定しきい値法

◆ 固定しきい値で二値化（ex: Threshold=100）

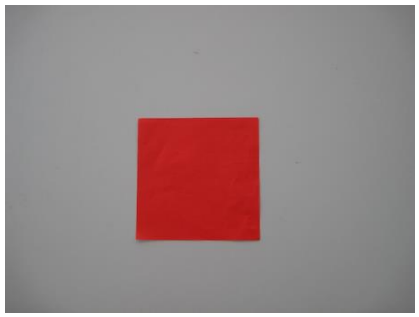
環境が変わらないことが前提

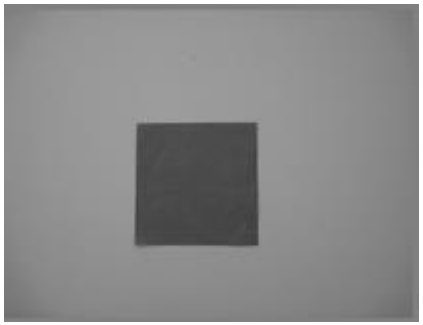
過去の演習問題

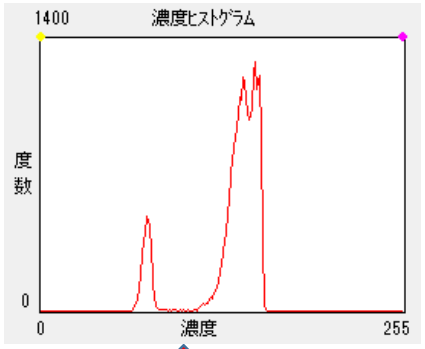
◆ 第4回 ヒストグラム

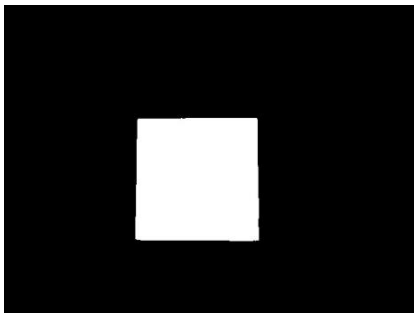
・ヒストグラムを作成して、しきいの見当をつける

・固定しきい値（t=100 ぐらい）を使用して二値化









100ぐらい？

二値化処理：固定しきい値法

```
    #define TH (100) //閾値
(略)
    //出力画像の宣言
    cv::Mat dst_img;
    dst_img.create(src_img.size(), src_img.type()); //領域確保
    //二値化
    for (int y=0; y<src_img.rows; y++) {
        for (int x=0; x<src_img.cols; x++) {
            if (src_img.at<unsigned char>(y, x) > TH) {
                dst_img.at<unsigned char>(y, x) = 255;
            }else{
                dst_img.at<unsigned char>(y, x) = 0;
            }
        }
    }
```

固定しきい値法　OpenCV の関数利用

```cpp
#include <iostream>
#include <opencv2/opencv.hpp>
#define FILE_NAME "red_rectangle.jpg"
//ウィンドウ名
#define WINDOW_NAME_INPUT "input"
#define WINDOW_NAME_OUTPUT "output"
//閾値
#define TH (100)
int main(int argc, const char * argv[]) {
    //画像をグレースケールで入力
    cv::Mat src_img = cv::imread(FILE_NAME, 0);
    if (src_img.empty()) { //入力失敗の場合
        fprintf(stderr, "File is not opened.\n");
        return (-1);
    }
    cv::Mat dst_img; //出力画像
    //二値化
    cv::threshold(src_img, dst_img, TH, 255, cv::THRESH_BINARY);
    //出力
    cv::imshow(WINDOW_NAME_INPUT, src_img);//画像の表示
    cv::imshow(WINDOW_NAME_OUTPUT, dst_img);//画像の表示
    cv::waitKey(); //キー入力待ち（止める）
    return 0;
}
```

```cpp
for (int y=0; y<src_img.rows; y++) {
    for (int x=0; x<src_img.cols; x++) {
        if (src_img.at<unsigned char>(y, x) > TH) {
            dst_img.at<unsigned char>(y, x) = 255;
        }else{
            dst_img.at<unsigned char>(y, x) = 0;
        }
    }
}
```

しきい値処理関数の説明

- 関数紹介

```
cv::threshold(入力画像, 出力画像, しきい値, max_value, オプション);
```

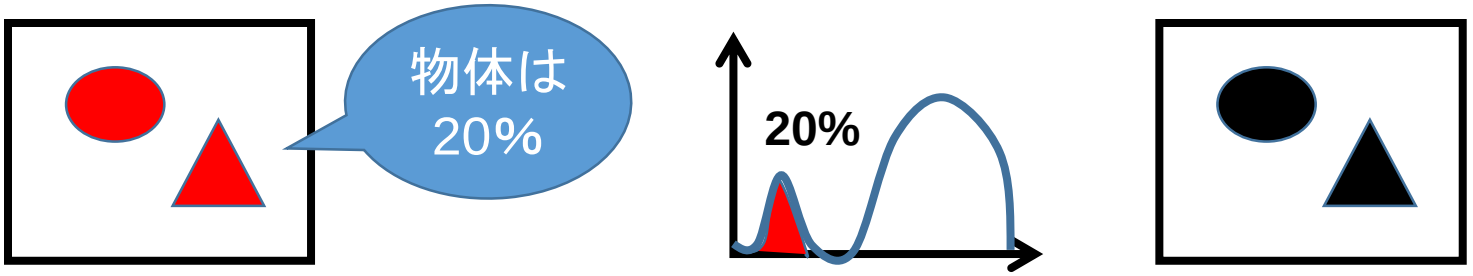
  - オプション
    - ➡ cv::THRESH_BINARY: しきい値以上をmax_valueに設定
    - ➡ cv::THRESH_BINARY_INV: しきい値以下をmax_value
    - ➡ など

```
//しきい値処理．しきい値以上を255にする
cv::threshold(src_img, dst_img, THRESHOLD, 255, cv::THRESH_BINARY);
```

しきい値の決定(探索)方法

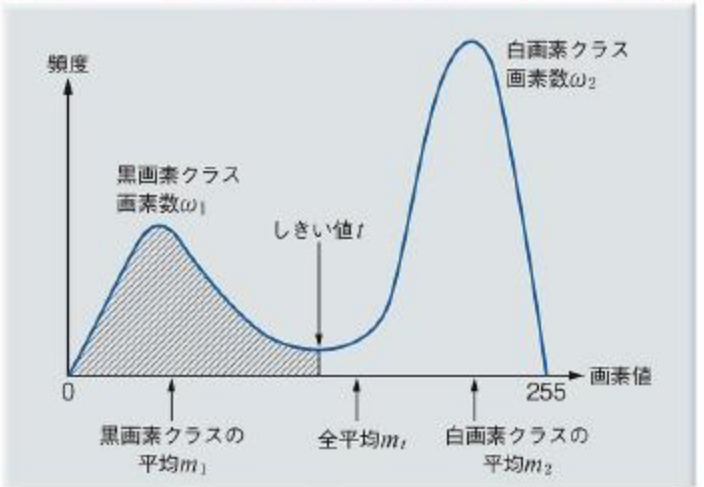
ノート

- P-タイル法 （第4回 チャレンジ課題）
  物体と背景の割り合いが既知



- 判別分析法(大津の判別分析法)
  物体と背景の2つに
  分かれることが既知

■図9.4——判別分析法によるしきい値 t の決め方

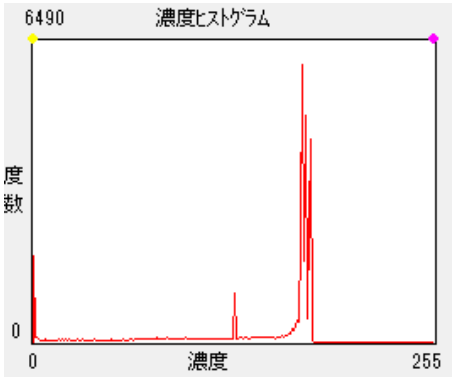


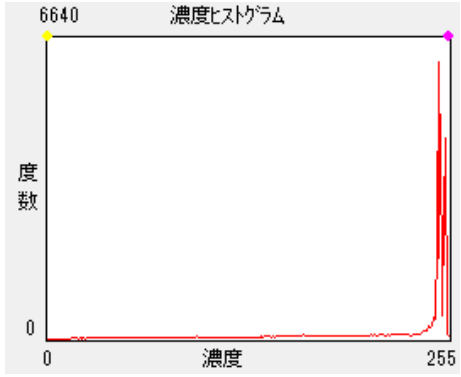
「ディジタル画像処理　改訂新版」2015年／公益財団法人 画像情報教育振興協会(CG-ARTS協会)

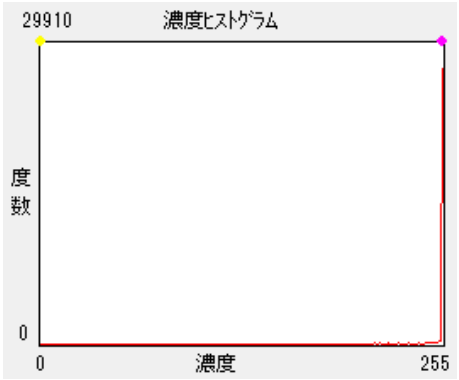
P–タイル法

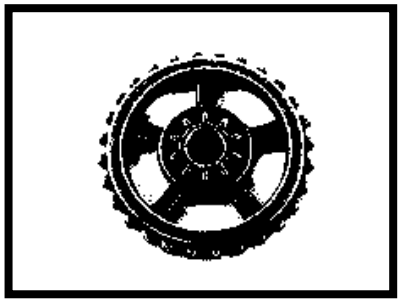
画像に占める物体の割合(P%)が判っている場合

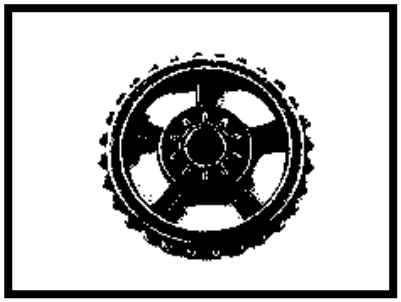
例：照明条件が変化する環境での部品切り出し

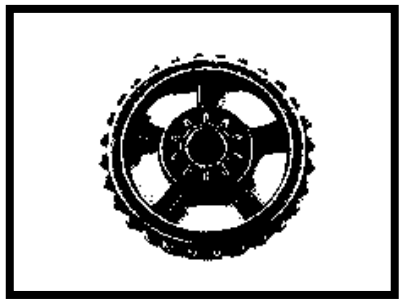






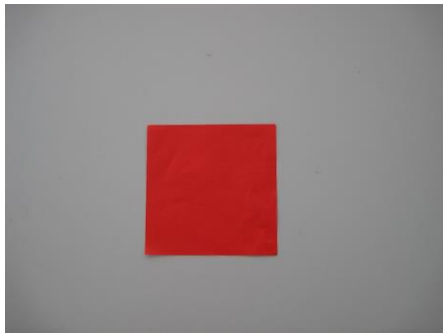


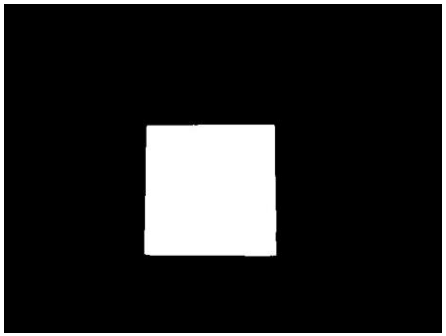


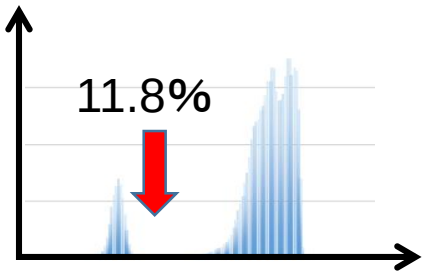


p-タイル法による物体と背景の切り分け

P-タイル法（教科書p.61）
rectangle.jpgをグレースケールで読み込み，
画素値が小さい方から数えて全体の11.8%の画素
を物体(白)，残りを背景(黒)として，二値化せよ．
プログラムで11.8という記載が必要

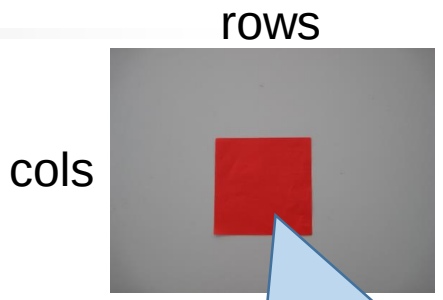




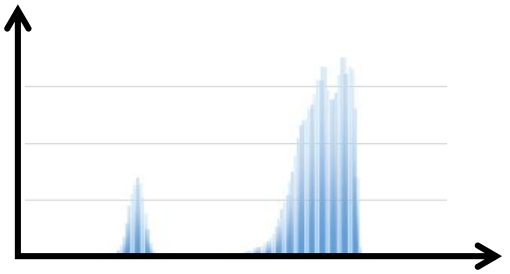


P–タイル法課題のヒント

1. 変数の宣言
2. 画像 (グレースケール) の読み込み
3. 出力画像のメモリ確保 (グレー)
4. Pタイル法で11.8%に相当する画素数を求める.
5. ヒストグラム用配列の初期化
6. ヒストグラムの生成 (計算)
7. 階級(閾値)の算出
8. 二値化
9. 表示



cols × rows × 11.8/100



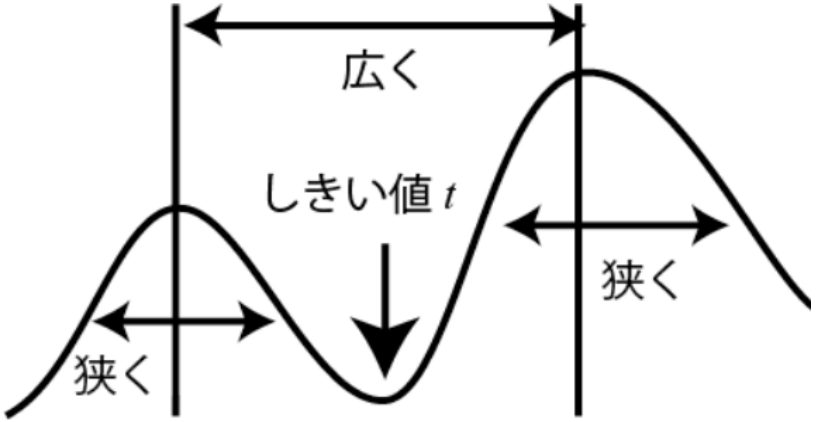
Σ h(0)+h(1)+h(2)+、、 h(Th) >= Num

判別分析法

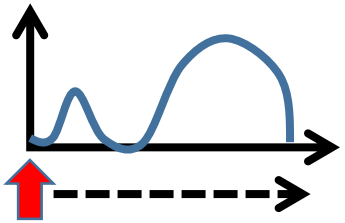
ノート

- クラス内 (山) の画素の値はほぼ同じ
- クラス間の画素値は離れる

$$\frac{\text{クラス間分散}}{\text{クラス内分散}} = \text{最大}$$



- 大津の判別分析法として有名



総当たり的

実際のアルゴリズム

しきい値　を　0から255まで順番に変えていき、それぞれの場合の
クラス内分散とクラス間分散を計算して、その比が最大の時を抽出

大津, "判別および最小２乗基準に基づく自動しきい値選定法",
電子通信学会論文誌, Vol.J63-D, No.4, pp.349-356, 1980.

演習　判別分析法

- 判別分析法による二値化
- プロジェクト名: otsu
- プロジェクトfixThのソースをコピーすること
- しきい値を以下のように設定 (1行のみ変更)

```
cv::threshold(src_img, dst_img, 0, 255,
      cv::THRESH_BINARY | cv::THRESH_OTSU);
```

二値画像処理（ここまでのまとめ）

ノート

- 二値化処理
  - 各画素を明るい画素と暗い画素に二分類
    - ➡ 白（255）もしくは黒（0）に変換
      - 固定しきい値法
      - pタイル法
      - 判別分析法
  - 輝度画像から二値画像を作成

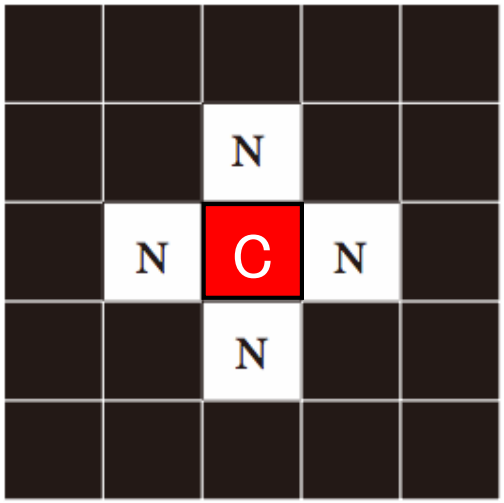
二値画像処理の基本

ノート

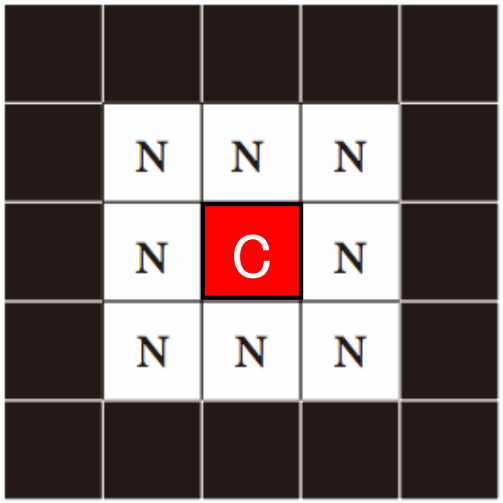
連結性 （教科書 p.64）

- つながっている画素の基準
- 斜めに位置する画素を
  近傍（neighbor）＝つながっている
  とみなすかどうか

4連結



8連結

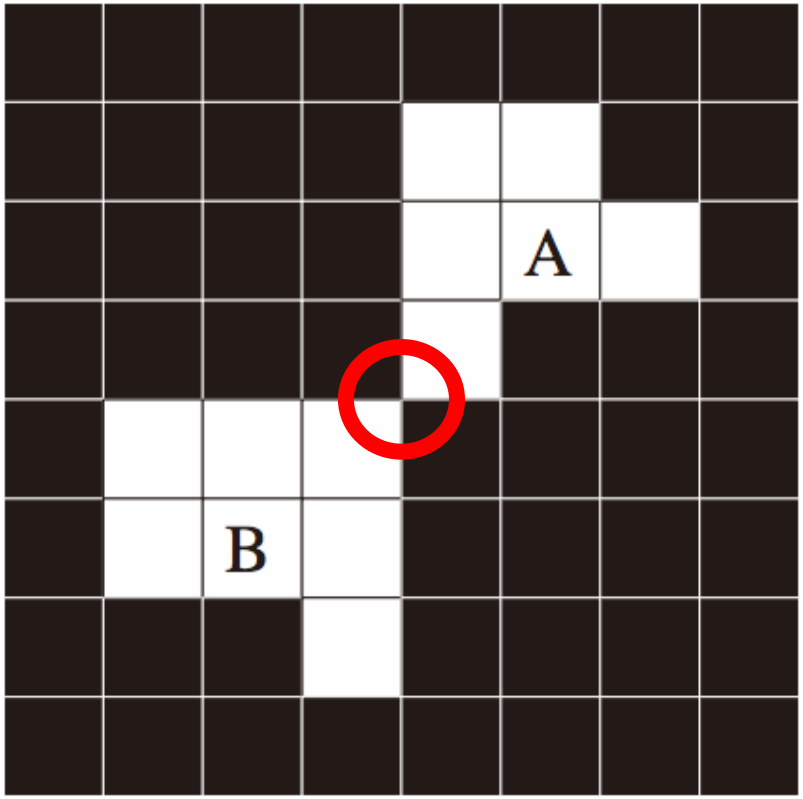


C: 注目画素に対し N: 近傍画素

左のNは４近傍　　右のNは８近傍

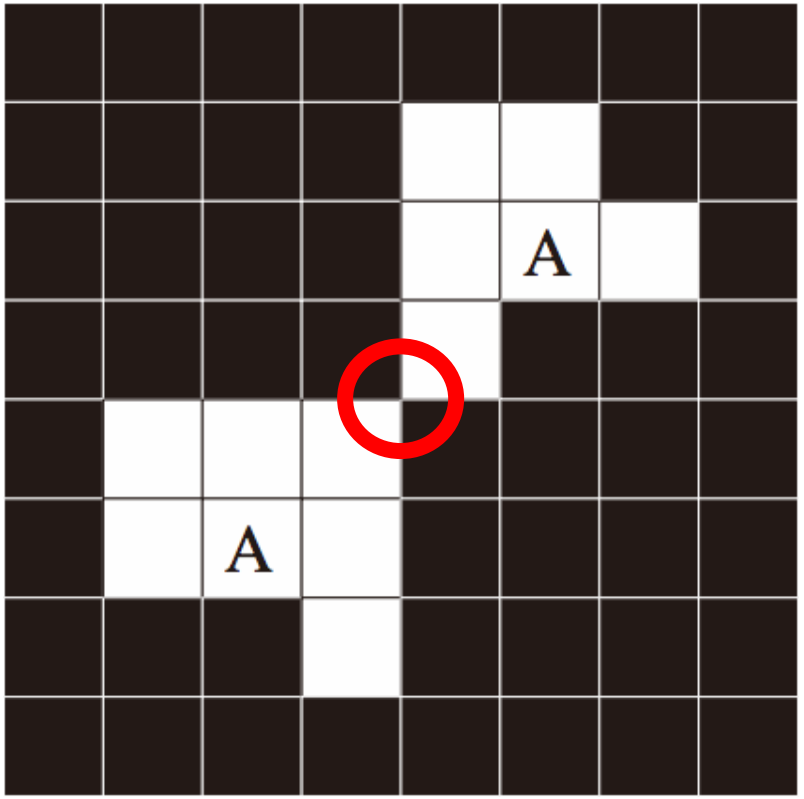
連結性による同一領域の違い

4連結



2つの領域となる
つながっていない

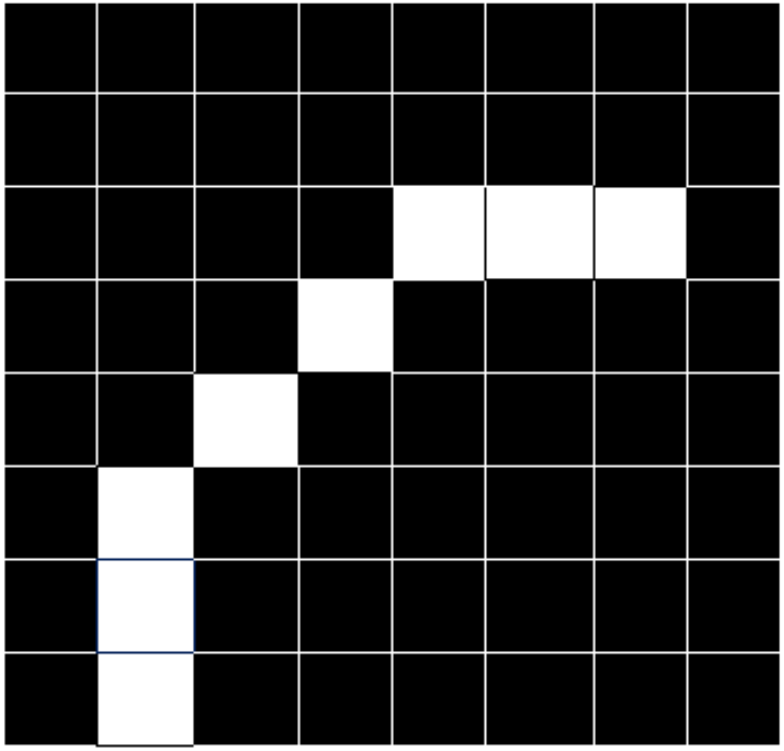
8連結



1つの領域となる
つながっている

二値画像処理

◆ この線はつながっている？ 領域はいくつ？



4連結
つながっていない
4つの領域

8連結
つながっている
1つの領域

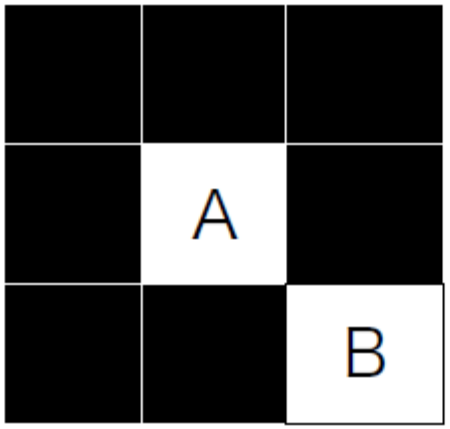
連結性によって変わる ⇒　輪郭追跡で重要

幾何学的性質

ノート

- A点(i, j)とB点(k, h)の距離
- ユークリッド距離 $\sqrt{(i-k)^2+(j-h)^2}$
- 4近傍距離 $|i-k|+|j-h|$
- 8近傍距離 $\max(|i-k|,|j-h|)$

例



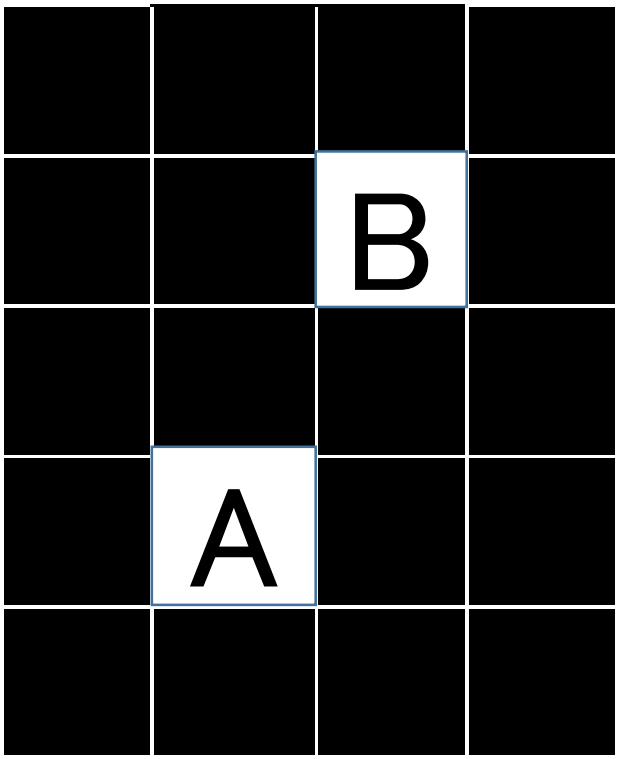
ユークリッド距離：$\sqrt{2}$

4近傍距離：2

8近傍距離：1

幾何学的性質

◆ A-B間の距離は？



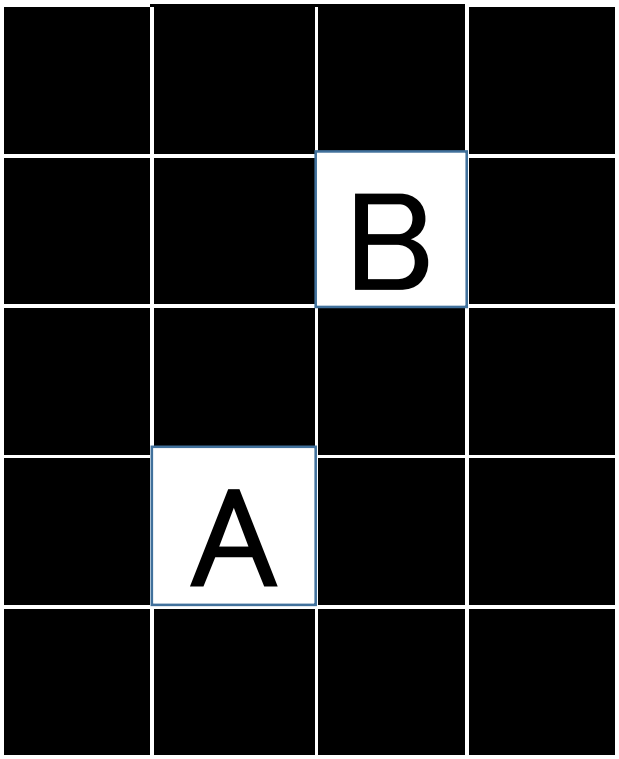
ユークリッド距離 ⇒　？

4近傍距離 ⇒　？

8近傍距離 ⇒　？

幾何学的性質

◆ A-B間の距離は？



ユークリッド距離 ⇒ $\sqrt{5}$

4近傍距離 ⇒ 3

8近傍距離 ⇒ 2

二値画像処理

ノート

◆ 二値化された画像に対する処理

- 画像から位置・形状情報を取り出すための処理
  - ➡ 膨張収縮処理
  - ➡ ラベリング処理
  - ➡ 輪郭追跡処理
  - ➡ 領域特徴量抽出
- 画素の連結性に基づいて処理

膨張・収縮処理 （教科書 p.65）

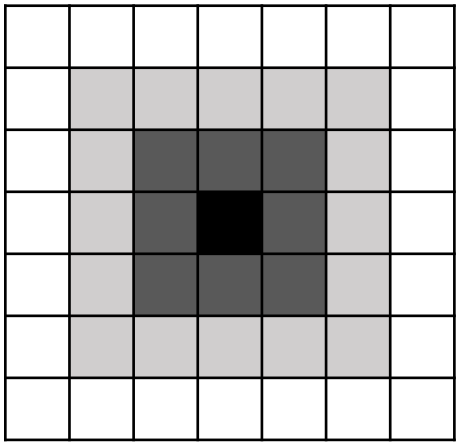
ノート

◆ 膨張処理

背景または穴に接する対象の画素に
画素をひと回り加える処理

◆ 収縮処理

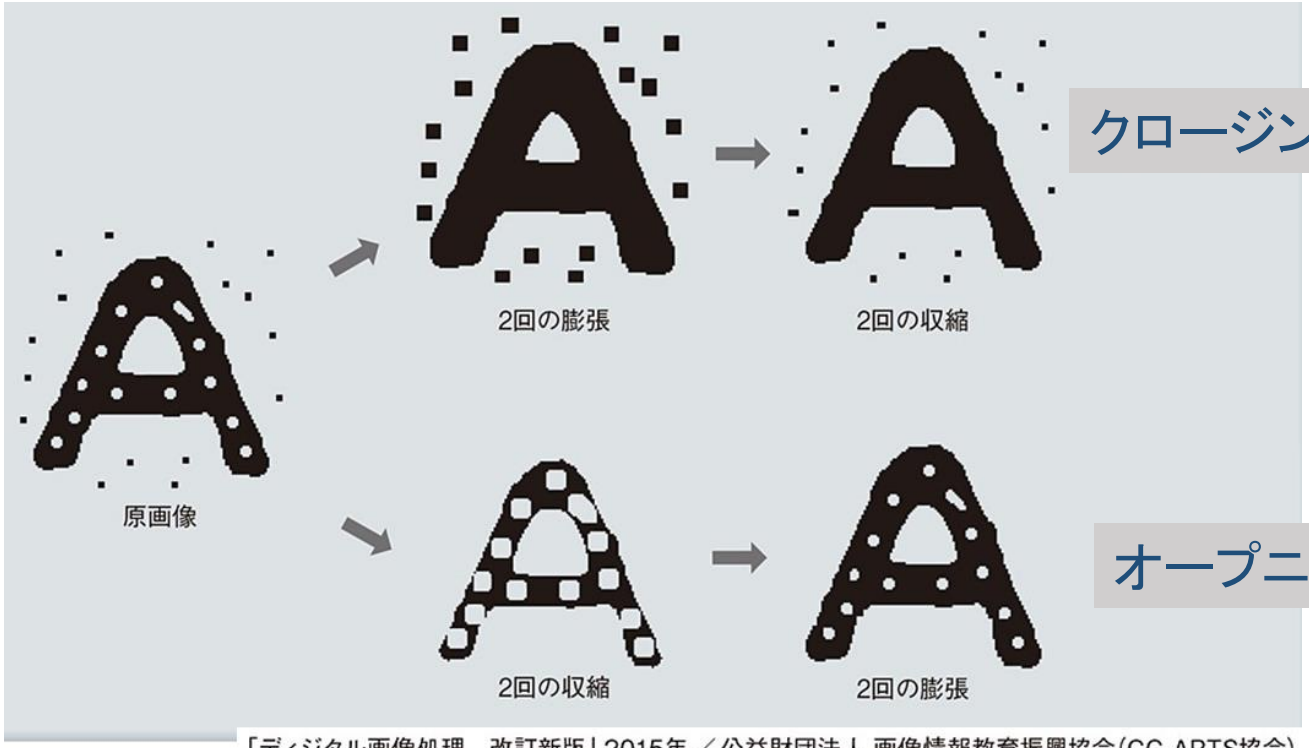
背景または穴に接する対象の画素に
画素をひと回り削る処理



白：抽出対象、黒；背景
白：背景、黒：抽出対象

クロージングとオープニング

- クロージング: 同じ回数だけ膨張して収縮
  ⇒ 画像の小さな穴を除去できる
- オープニング: 同じ処理だけ収縮して膨張
  ⇒ 画像の小さな連結成分を除去できる



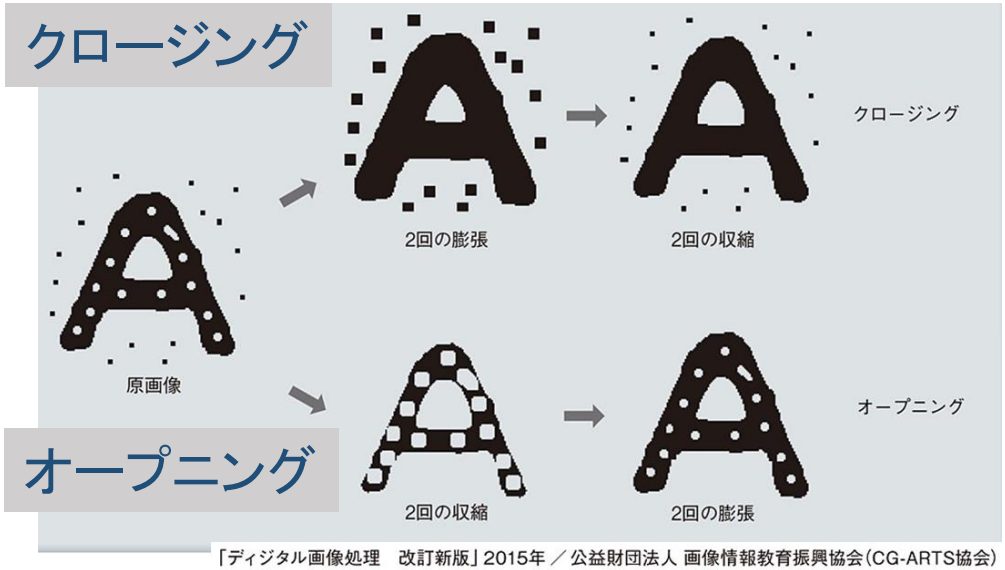
「ディジタル画像処理　改訂新版」2015年 ／ 公益財団法人 画像情報教育振興協会(CG-ARTS協会)

クロージングとオープニング 注意事項

① オープニングとクロージングの処理の順序を入れ替えた場合，同じ結果であるとは限らない

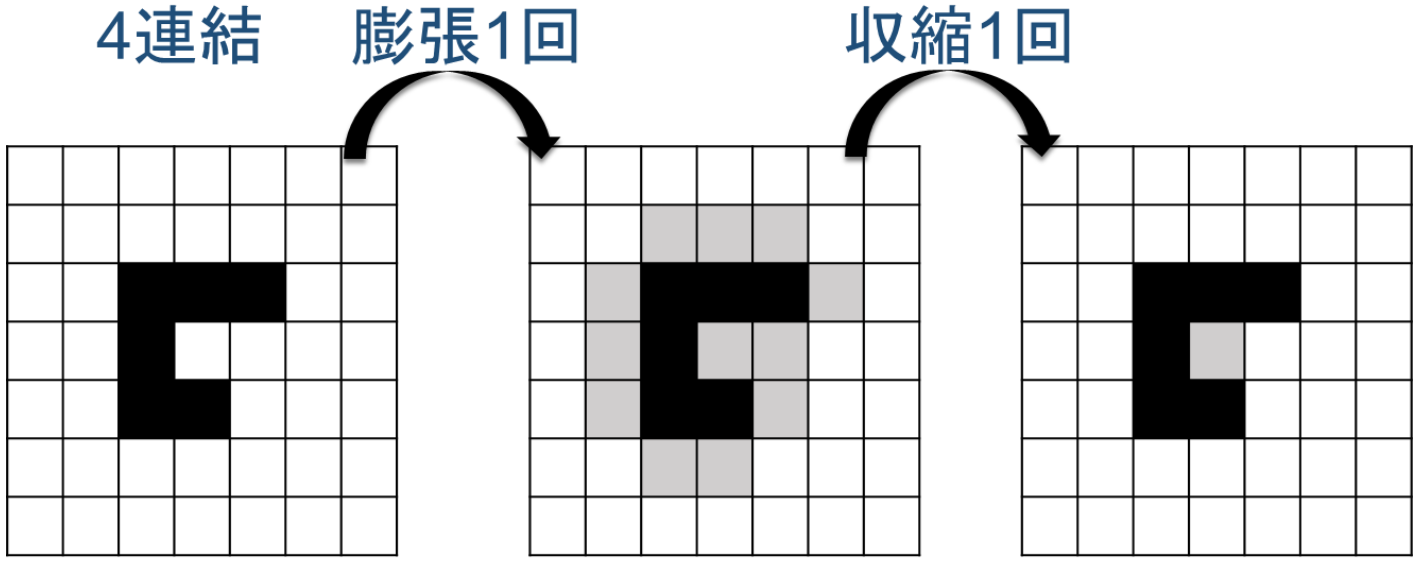
② 膨張・収縮の回数は、どれくらいの大きさの穴と連結成分をノイズとして削除したいかに応じて決める

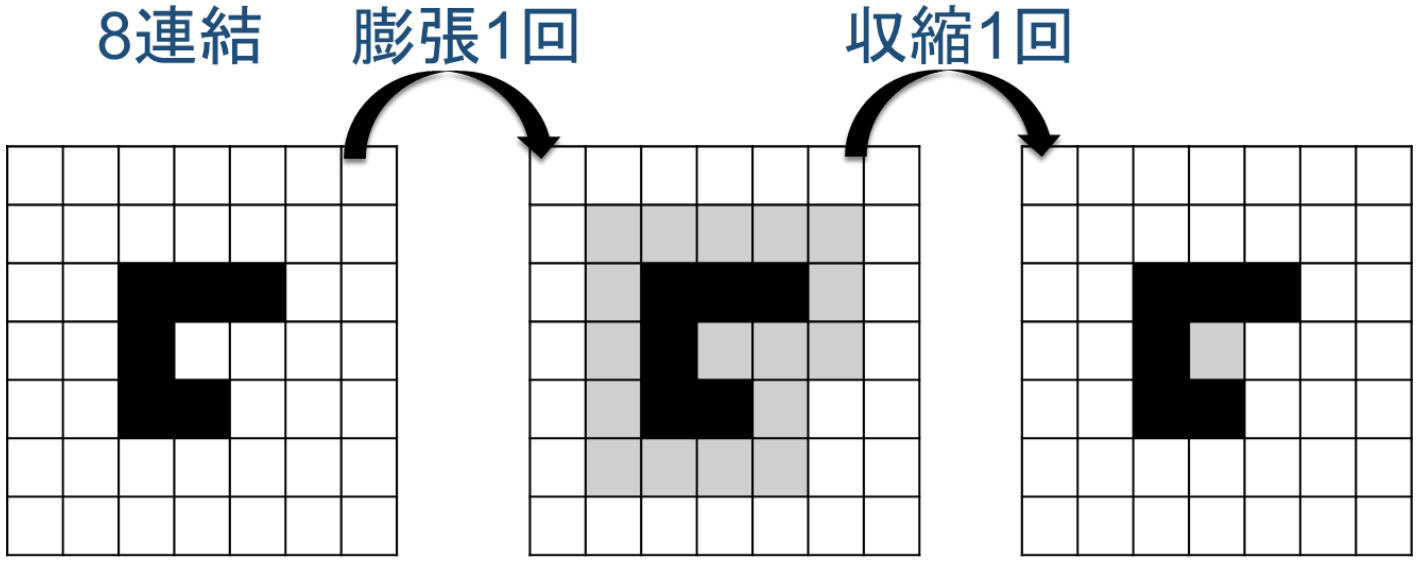
③ 連結性（4連結、8連結）による結果の違いに注意



「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)

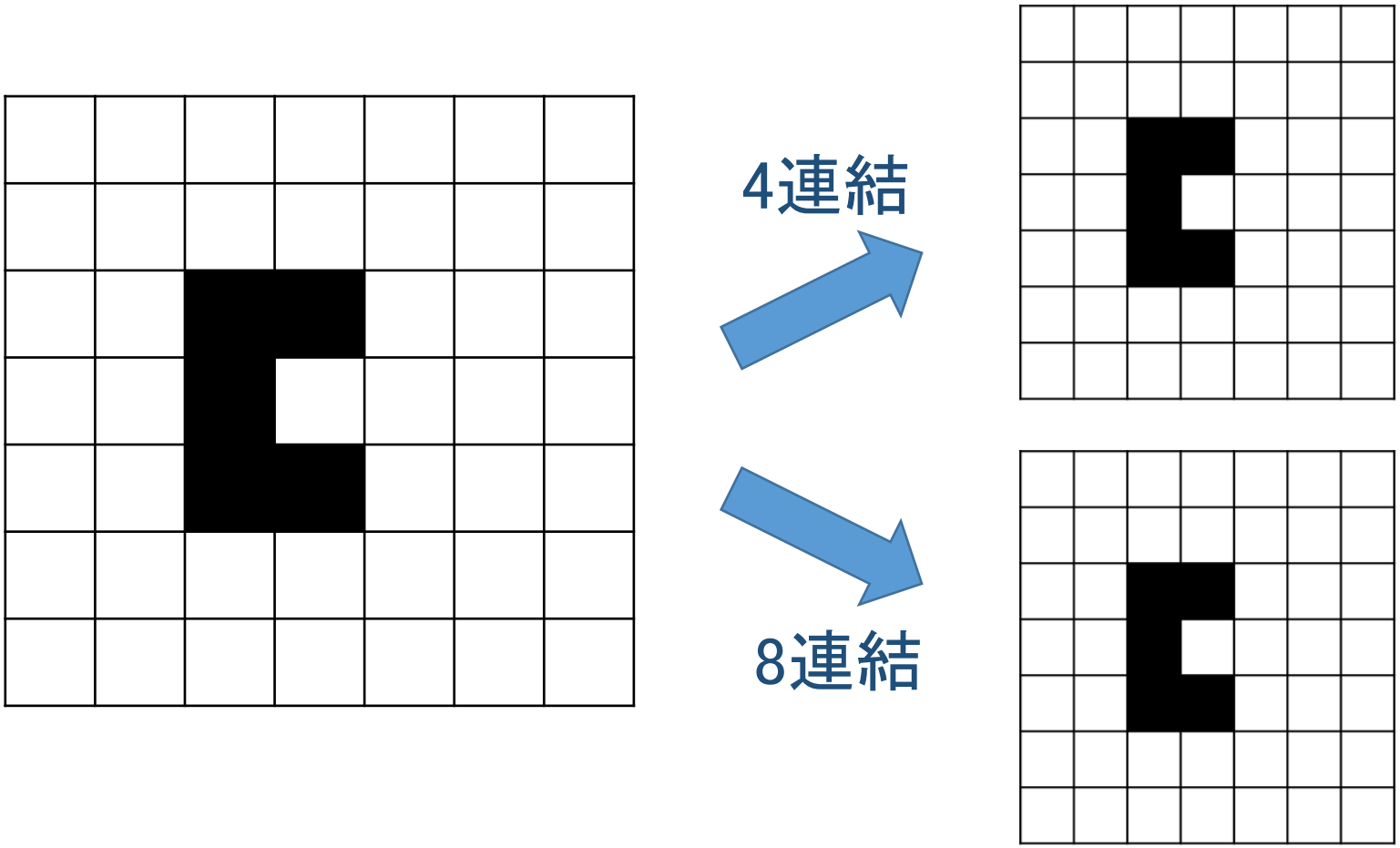
連結性（4連結、8連結）による結果の違い





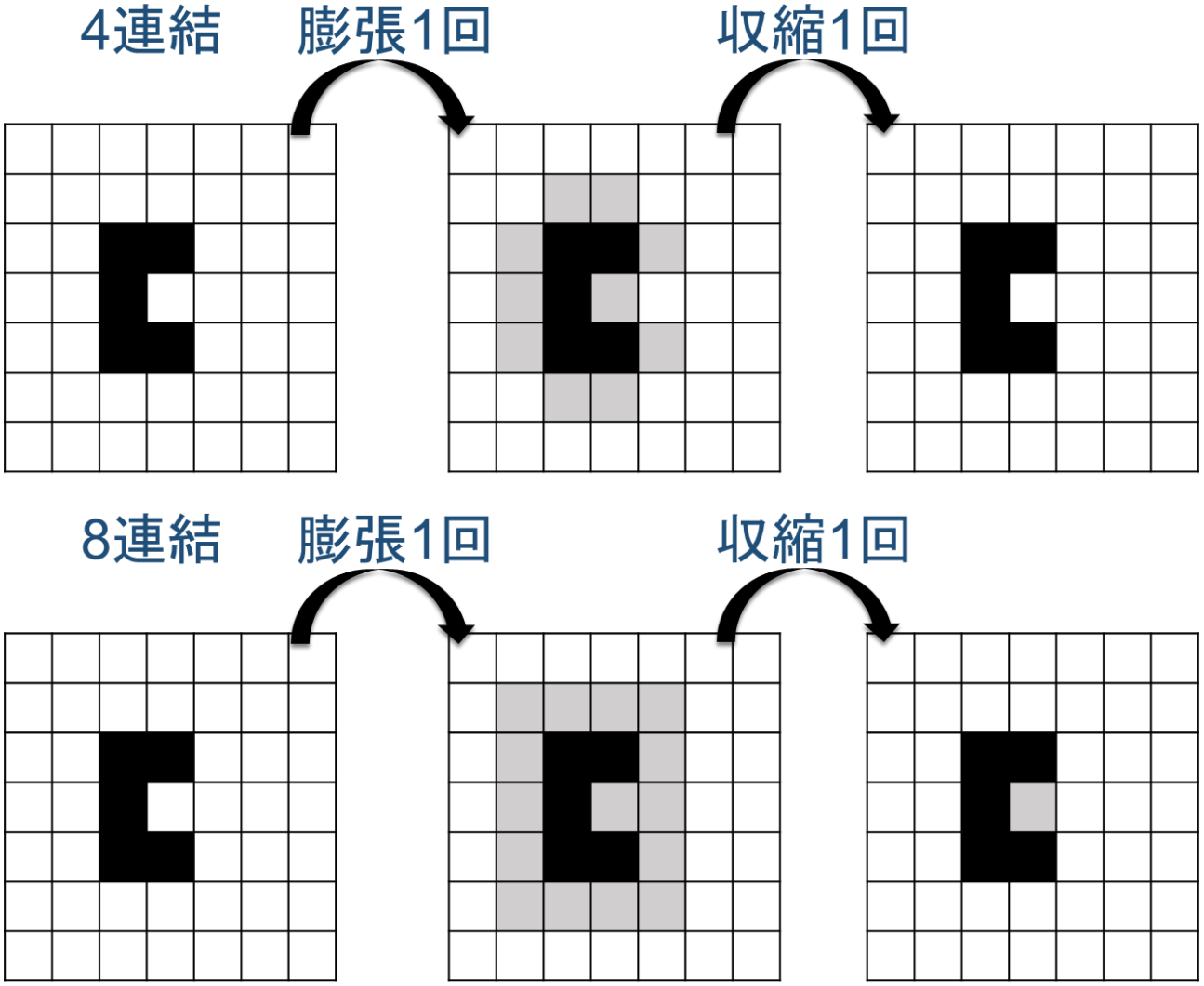
【確認】連結性（4連結、8連結）による結果の違い

◆ 4/8連結で膨張1回、収縮1回実施後の結果を求めよ



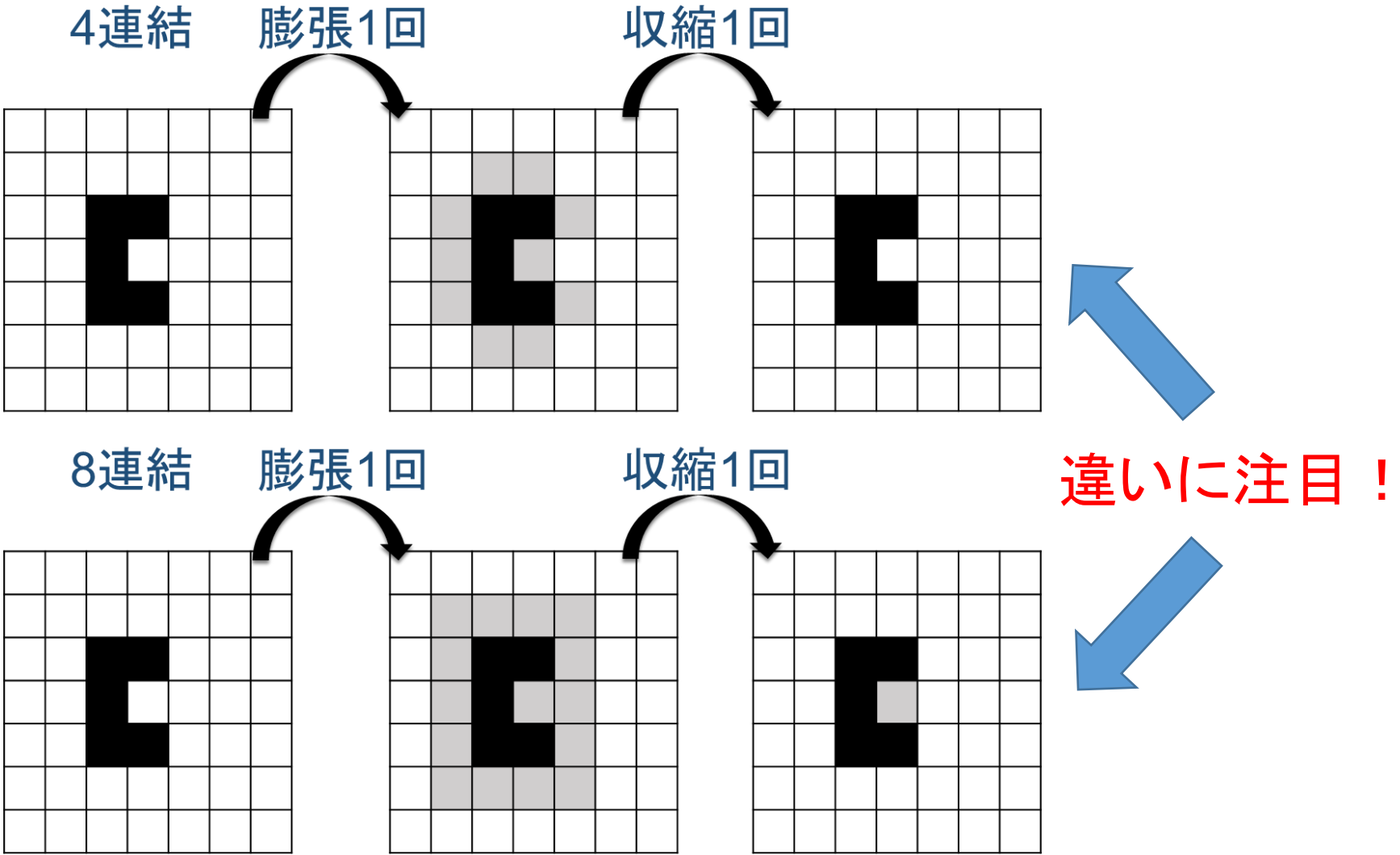
【確認】連結性（4連結、8連結）による結果の違い

◆ 4/8連結で膨張1回、収縮1回実施後の結果を求めよ



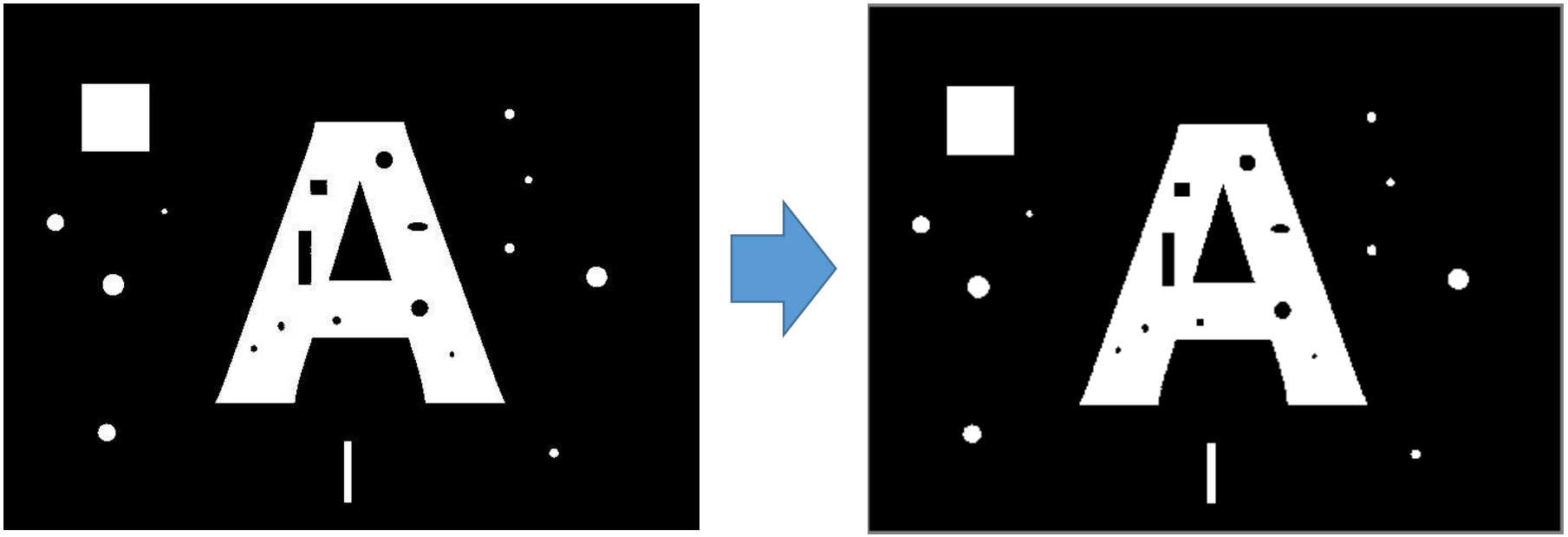
【演習】連結性（4連結、8連結）による結果の違い

◆ 4/8連結で膨張1回、収縮1回実施後の結果を求めよ



【演習】プロジェクト名 ： morphology

- 内容：膨張・収縮処理の実践　OpenCV の関数利用
  入力画像（a_img.jpg）　[2値画像に見えるけど実は濃淡画像]
- 事前準備
  ① グレースケールで画像を入力
  ② 二値化（固定閾値: 100）
  ③ 表示



膨張処理用の関数 （OpenCV）

- 入力画像src_imgに対して膨張処理を行う

```
cv::dilate(入力画像, 出力画像, 構造要素, 位置, 回数);
```

- 構造要素にcv::Mat()を入れることで3x3の矩形構造要素が使用される
- 位置は構造要素の中心

```
cv::dilate(src_img, dst_img, cv::Mat(), cv::Point(-1, -1), 10);
```

収縮処理用の関数 （OpenCV）

・入力画像src_imgに対して膨張処理を行う

```
cv::erode(入力画像, 出力画像, 構造要素, 位置, 回数);
```

・構造要素にcv::Mat()を入れることで3x3の
矩形構造要素が使用される

・位置は構造要素の中心

```
cv::erode(bin_img, dst_img, cv::Mat(), cv::Point(-1, -1), 10);
```

【演習】10回膨張させてみよう

- キー入力で画像の変化を確認する

```
  //膨張
for (i=0; i<10; i++) {
  //膨張
  cv::dilate(bin_img, dst_img, cv::Mat(), cv::Point(-1, -1), i);
  cv::imshow(WINDOW_NAME_OUTPUT, dst_img); //表示
  cv::waitKey(); //キー入力待ち
}
```

- その後，10回収縮し，元の画像と比較すること
- 膨張と収縮の順番を入れ替えてみたら？

【演習】で確認してもらいたいこと
① 10回膨張　②10回膨張、10回収縮　③ 10回収縮、10回膨張