OpenCV for C++ による画像処理(5)

1. 画像の 2 値化

画像の画素値を，ある値（閾値）との大小関係によって 0 か 255（0 か 1）に変換する処理を 2 値化という．

以下に，2 値化の基準となる閾値をトラックバーを使って指定して 2 値化画像を得るプログラムを示す．

```
1:  #include "pch.h"
2:  #include <opencv2¥opencv.hpp>
3:
4:  #pragma comment(lib, "opencv_world347d.lib")
5:
6:  using namespace std;
7:  using namespace cv;
8:
9:  int th = 127;
10: void changeThreshold(int position, void* userdata);  //トラックバーコールバック関数
11:
12: int main()
13: {
14:     Mat src = imread("G:¥¥yama¥¥sidba¥¥text.bmp");
15:     Mat gray, dst;
16:
17:     imshow("Input image", src);
18:
19:     cvtColor(src, gray, COLOR_BGR2GRAY); //入力画像のグレースケール化
20:     imshow("Gray image", gray);
21:
22:     //トラックバーの設置
23:     namedWindow("Binary Image");
24:     createTrackbar("th", "Binary Image", &th, 255, changeThreshold);
25:
26:     while (1) {
27:         threshold(gray, dst, th, 255, THRESH_BINARY);  //2値化処理（閾値th）
28:         imshow("Binary Image", dst);
29:
30:         if (waitKey(30) == 27) break;
31:     }
32:
33:     return 0;
34: }
35:
36: //トラックバーコールバック関数（閾値の変更）
37: void changeThreshold(int position, void* userdata)
38: {
39:     th = position;  //トラックバーの位置のよって閾値を変更
40: }
```

トラックバーによって閾値を指定する 2 値化処理プログラム

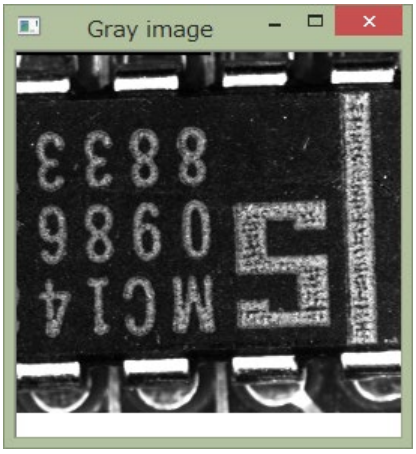
このプログラムでは，入力画像 src を一度グレースケール化し，その画像 gray に対して 2 値化処理を施すようにしている．

画像表示ウィンドウ”Binary Image”にトラックバーを設置し，26～31 行目の while(1)無限ループの中でトラックバーの位置 th を読み取りつつ threshold()関数で 2 値化している．

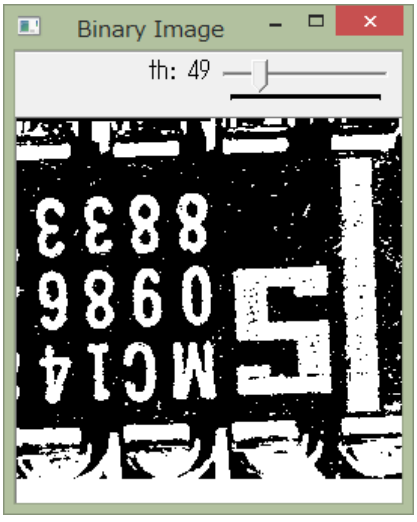
```
threshold(gray, dst, th, 255, THRESH_BINARY);
```

threshold()関数では，第 1 引数に 2 値化の対象となるグレースケール画像（1 チャネル画像）を指定する．第 2 引数に結果画像，第 3 引数に閾値を指定する．
第 4 引数には 2 値化した後の値の最大値を指定（基本的には 255：白を指定，最小値は自動的に 0：黒となる）する．
第 5 引数については以下の表のような 2 値化の処理モードを指定する．

THESH_BINARY	閾値より大きい値を最大値，以下の値を 0 に
THRESH_BINARU_INV	閾値より大きい値を 0，以下の値を最大値に
THRESH_TRUNC	閾値より大きい値を閾値に，以下の値を 0 に
THRESH_TOZERO	閾値より大きい値をそのまま（画素値），以下の値を 0 に
THRESH_TOZERO_INV	閾値より大きい値を 0 に，以下の値をそのまま（画素値）



(a) 入力画像をグレースケール化



(b) 閾値 49 で 2 値化

実行結果

2．2 値画像に対する膨張・収縮処理(Dilation, Erosion)

上述のプログラムの処理結果(b)を見ると，文字部分の内部に黒い画素が含まれていたり，背景部分には白い画素が含まれていたりするのが見える．このようなノイズを除去するために，膨張（Dilation）・収縮（Erosion）処理が施される．

以下に，トラックバーによって膨張，収縮処理を施す回数を指定して，それらの結果を表示するプログラムを示す（上部の include，using などは省略）．

```
1:  int di_ite = 0;
2:  void changeDilrateIte(int position, void* userdata);  //トラックバーコールバック関数
3:  int er_ite = 0;
4:  void changeErodeIte(int position, void* userdata);  //トラックバーコールバック関数
5:
6:  int main()
7:  {
8:      Mat src = imread("G:¥¥yama¥¥sidba¥¥text.bmp");
9:      Mat gray, binimg;
10:     Mat di_dst, er_dst;
11:
12:     imshow("Input image", src);
13:
14:     cvtColor(src, gray, COLOR_BGR2GRAY);  //入力画像をグレースケール化
15:     threshold(gray, binimg, 127, 255, THRESH_BINARY);  //2値化処理
16:     imshow("Binary image", binimg);
17:
18:     //トラックバーの設置（Dilate回数）
19:     namedWindow("Dilate Image");
20:     createTrackbar("Dilate", "Dilate Image", &di_ite, 10, changeDilrateIte);
21:     //トラックバーの設置（Erode回数）
22:     namedWindow("Erode Image");
23:     createTrackbar("Erode", "Erode Image", &er_ite, 10, changeErodeIte);
24:
25:     while (1) {
26:         dilate(binimg, di_dst, noArray(), Point(-1, -1), di_ite);  //Dilate処理
27:         imshow("Dilate Image", di_dst);
28:
29:         erode(binimg, er_dst, noArray(), Point(-1, -1), er_ite);  //Erode処理
30:         imshow("Erode Image", er_dst);
31:
32:         if (waitKey(30) == 27) break;
33:     }
34:
35:     return 0;
36: }
37:
38: //トラックバーコールバック関数（Dilateの回数）
39: void changeDilrateIte(int position, void* userdata)
40: {
41:     di_ite = position;  //トラックバーの位置のよってDilate回数を変更
42: }
43:
44: //トラックバーコールバック関数（Erodeの回数）
45: void changeErodeIte(int position, void* userdata)
46: {
47:     er_ite = position;  //トラックバーの位置のよってErode回数を変更
48: }
```

2 値画像に対する膨張と収縮処理プログラム（トラックバーで処理回数指定）

このプログラムでは，入力画像 src を一度グレースケール画像 gray に変換し，閾値 127 で 2 値化した画像 binimg に対して膨張,収縮処理を施すようにしている.また,それらの処理回数 dr_ite, er_ite はトラックバーで指定するようにしている．それぞれ，初期値を 0 回，最大回数を 10 回として設定している.

膨張処理，収縮処理は

```
dilate(binimg, di_dst, noArray(), Point(-1, -1), di_ite);
erode(binimg, er_dst, noArray(), Point(-1, -1), er_ite);
```

の関数で施される.
第 1 引数に入力画像の cv::Mat データ，第 2 引数に結果を格納する cv::Mat データを指定する.
第 3 引数には，膨張・収縮処理を施すときの処理判定（注目画素を 255 とするか，0 とするか）の近傍範囲を cv::Mat データで指定する．noArray()と指定すると，デフォルトの 3×3 画素領域での判定を行う.
第 4 引数には判定領域のどの画素を 255 もしくは 0 にするかを指定する．Point(-1,-1)とすると，中心画素を変更することになる.
第 5 引数に膨張，収縮処理を施す回数を指定する.

(a) 膨張処理 2 回の結果　　(b) 収縮処理 2 回の結果

膨張･収縮処理の結果

3. 2 値画像に対するラベリング処理

ある対象物が写っている画像を 2 値化し，その 2 値画像上で白画素が連結している図形成分を認識することによって，対象物の形や大きさなどが解析できる．図形成分を認識する処理としてラベリング処理がある．これは，図形成分を構成する白画素ひとつひとつに同じラベルを張り付けていく処理である.

```
1:  int main()
2:  {
3:      Mat src = imread("G:¥¥yama¥¥sidba¥¥text.bmp");
4:      Mat gray, binimg, labimg;
5:
6:      imshow("Input image", src);
7:
8:      cvtColor(src, gray, COLOR_BGR2GRAY);  //入力画像をグレースケール化
9:
10:     threshold(gray, binimg, 127, 255, THRESH_BINARY);  //2値化処理
11:
12:     dilate(binimg, binimg, noArray(), Point(-1, -1), 2);  //2値化した画像に2回Dilate処
13: 理
14:     imshow("Binary-Dilate Image", binimg);
15:
16:     int nLabs = connectedComponents(binimg, labimg, 8, CV_32S);  //ラベリング処理
17:
18:     // ラベリング結果の描画色を決定
19:     vector<Vec3b> colors(nLabs);
20:     colors[0] = Vec3b(0, 0, 0);
21:     for (int label = 1; label < nLabs; ++label) {
22:         //ラベル番号に対して色をランダムに割り当てる
23:         colors[label] = Vec3b((rand() & 255), (rand() & 255), (rand() & 255));
24:     }
25:
26:     // ラベリング結果の描画
27:     Mat dst(src.size(), CV_8UC3);
28:     for (int y = 0; y < dst.rows; ++y) {
29:         for (int x = 0; x < dst.cols; ++x) {
30:             //ラベリング画像の(x,y)上のラベル番号を抽出
31:             int label = labimg.at<int>(y, x);
32:             //ラベル番号に割り当てられた色（画素値）を結果画像の(x,y)に格納する
33:             cv::Vec3b &pixel = dst.at<cv::Vec3b>(y, x);
34:             pixel = colors[label];
35:         }
36:     }
37:     imshow("Labeling Image", dst);
38:
39:     waitKey();
40:
41:     return 0;
42: }
```

2 値画像に対するラベリング処理プログラム

このプログラムでは，入力画像 src を一度グレースケール画像 gray に変換し，閾値 127 で 2 値化，さらに膨張処理を 2 回施した画像 binimg に対してラベリング処理を施すようにしている．

ラベリング処理は connectedComponents()関数で行っている．この関数が実行されると，ラベリングされた図形成分の個数が返ってくる（プログラムでは nLabs に格納している）．

```
int nLabs = connectedCompponents(binimg, labimg, 8, CV_32S);
```

第 1 引数にラベリングする 2 値画像（cv:Mat データ，1 チャネル），第 2 引数にラベリング結果

の画像の cv:Mat データを指定する．ラベリング結果の画像のデータ型は第 4 引数で指定する．個の場合は，int 型（4 バイト整数）で指定している．

第 3 引数は，注目している白画素が，他の白画素と連結しているかどうかを判定するときの範囲を指定している．8 近傍であれば 8，4 近傍であれば 4 を指定する．

18～36 行目では，ラベリング結果画像 labimg を基に，各図形成分を色付きで表示するための画像 dst を作成している．19～24 行目で各ラベル番号をどの色で表示するかをランダムに決定し，27～36 行目でラベルの付いた画素に対してその番号に応じた色を格納することによって表示画像を作成している．

(a) 2 値化後，膨張処理 2 回施した画像　　(b) ラベリング結果

ラベリング処理の結果