

openCV レポート 2 ~特徴点抽出~

平成 20 年 2 月 15 日

1 はじめに

今日の画像処理、例えば 3 次元復元などは画像から点や線などの情報を得ると言った射影幾何学などの解析に基づいている。

よってコンピュータでこのような画像処理の研究を行うためには、実際に画像上から特徴点と呼ばれるような点を抽出する必要がある。

そこで本稿では画像から特徴点を抽出する方法の一例としてオープンソースの画像処理ライブラリである openCV の関数 cvGoodFeaturesToTrack を用いる方法について述べる。

2 cvGoodFeaturesToTrack 関数

cvGoodFeaturesToTrack 関数は米 intel 社が提供するオープンソースの画像処理ライブラリである openCV の関数であり、画像内の鮮明なコーナーを検出する。具体的には先ず、入力された画像の全てのピクセルに対して輝度値 $I(x, y)$ を求める。

$$I(x, y) = f(x, y) \quad (1)$$

図 1a のように求めた輝度値をそのピクセルの高さと解釈し、 x および y を連続変数と見なせば、画像をひとつの曲面と捉えることができる。(図 1b)

入力画像の全てのピクセルに対して、注目するピクセルの $block-size \times block-size$ の隣接領域(今回は 3×3 とした)の輝度値の頂点を結ぶ曲面の x 方向、 y 方向の微分値を f_x 、 f_y とする。(図 1c)

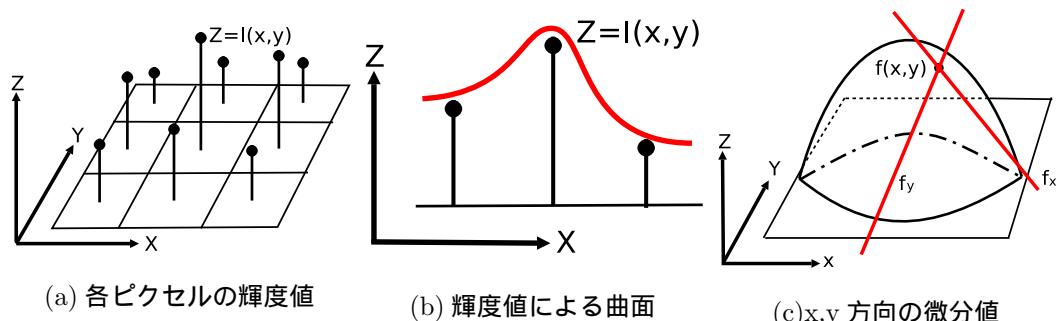


図 1: cvGoodFeaturesToTrack 関数概要

次に各ピクセルの固有値を求め、隣接領域内の固有値の中で極大の物のみを残す。
残った固有値の内で閾値(今回は openCV のサンプルと同じく 0.01 とした)以下の物を削除する。
最後に、この関数はコーナー点に着目し(最も強いコーナーが一番最初に対象となる)、新しく着目した特徴点とそれ以前に対象とした特徴点群との距離が閾値(今回は 5 ピクセル)よりも大きい

ことをチェックすることで、すべての検出されたコーナーそれぞれの距離が十分離れていることを保証する。そのため、この関数は鮮明な特徴点との距離が近い特徴点を削除する。

`cvGoodFeaturesToTrack` 関数では、コーナーの検出器として `cvCornerMinEigenVal` 関数と Harris 検出器のどちらかを使用することが可能である。次にこの 2 つの検出器について説明する。

2.1 `cvCornerMinEigenVal` 関数

`cvCornerMinEigenVal` 関数は先に述べた、注目ピクセルの $\text{block-size} \times \text{block-size}$ の隣接領域における微分値から (2) 式の行列 M を定義する。

$$M = \begin{pmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{pmatrix} \quad (2)$$

(2) 式の行列の最小固有値を求め、`cvGoodFeatureToTrack` 関数に値を返す。

2.2 Harris 検出器

Harris 検出器は今日の画像処理において、コーナー検出に広く用いられている特徴点抽出法である。openCV では、`cvCornerHarris` 関数がこれに当たる。微分幾何学に基づくアプローチでは 2 階微分が必要であるが、Harris 検出器は (3),(4) 式のように定義され、1 階微分のみしか計算しない。

$$Harris = \det \hat{C} - k(\text{tr } \hat{C}) = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (3)$$

(3) 式中の k は Harris 検出器における定数であり、Harris 自身は 0.04 を推奨しているが金澤、金谷らは実験的に 0.06 ~ 0.08 が適当であるとしており、今日では 0.06 の値が広く用いられている。 λ_1, λ_2 は (4) 式の固有値である。

$$\hat{C} = \begin{pmatrix} G_\sigma(f_x^2) & G_\sigma(f_x f_y) \\ G_\sigma(f_x f_y) & G_\sigma(f_y^2) \end{pmatrix} \quad (4)$$

ここで $G_\sigma()$ は標準偏差 σ のガウス分布による平滑化を表しており、openCVにおいては `cvSmooth` 関数がこれに当たる。なお標準偏差 σ は (5) 式によって計算され、今回のアルゴリズムでは 0.95 としている ($n = 3$)。

$$\sigma = \left(\frac{n}{2} - 1 \right) \times 0.3 + 0.8 \quad (5)$$

(5) 式中の n はガウシアンフィルタの範囲 ($n \times m$ 今回は 3×3) である。
入力画像の全てのピクセルに対して注目ピクセルの $\text{block-size} \times \text{block-size}$ の隣接領域において 2×2 サイズの勾配から (4) 式の行列を用いて固有値を計算し、求めた値から (3) 式を計算、`cvGoodFeaturesToTrack` 関数に値を返す。今回の特徴点抽出では、この Harris 検出器を用いている。

2.3 リファレンス

`cvGoodFeatureToTrack` 関数の引数などリファレンスを以下に示す。

```
void cvGoodFeaturesToTrack(const CvArr* image,
                           CvArr* eig_image,
                           CvArr* temp_image,
                           CvPoint2D32f* corners,
                           int* corner_count,
                           double quality_level,
```

```

        double min_distance,
        const CvArr* mask=NULL,
        int block_size=3,
        int use_harris=0,
        double k=0.06 );

```

ここで、関数の各引数は次の通りである。

```

image
8 ビット、または 32 ビット浮動小数点型シングルチャンネルの入力画像。
eig_image
32 ビット浮動小数点型のテンポラリ画像。サイズは image と同じである。
temp_image
別のテンポラリ画像。サイズ・フォーマットとともに eig_image と同じである。
corners
出力パラメータ。検出されたコーナー。
corner_count
出力パラメータ。検出されたコーナーの数。
quality_level
最大・最小の固有値に乘ずる定数（検出される画像コーナの許容最低品質を指定する）。
min_distance
出力される画像コーナー間の許容最低距離（ユークリッド距離を用いる）。
mask
注目領域（ROI）。指定された領域または mask が NULL の場合は画像全体から点を選ぶ。
block_size
平均化ブロックのサイズ。この関数で内部的に用いられる cvCornerMinEigenVal 関数 あるいは、cvCornerHarris 関数 に、この引数が渡される。
use_harris
0 でない場合は、デフォルトの cvCornerMinEigenVal 関数 の代わりに Harris 検出器（cvCornerHarris 関数）を用いる。
k
Harris 検出器のパラメータ（use_harris 0 のときのみ使用）。

```

3 プログラム

3.1 ソースコード

実際に cvGoodFeaturesToTrack 関数を用いた特徴点抽出のソースコードを以下に記す。

```

01 #include "cxcore.h"
02 #include "cv.h"
03 #include "highgui.h"
04
05 int main(int argc,char **argv)
06 {
07 /* 検出したコーナーを格納 */
08 CvPoint2D32f corners[1000];
09 /* 最大検出数 */
10 int count = 1000;
11
12 /* メモリ確保 */
13 //入力画像 (File.bmp)
14 IplImage *src = cvLoadImage("File.bmp",-1);

```

```

15 //入力画像（グレースケール）
16 IplImage *gray = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U,1);
17 //一時画像 (IPL_DEPTH_32F)
18 IplImage *eigen = cvCreateImage(cvGetSize(src), IPL_DEPTH_32F,1);
19 //一時画像 (IPL_DEPTH_32F)
20 IplImage *temp = cvCreateImage(cvGetSize(src), IPL_DEPTH_32F,1);
21 //結果画像 (検出したコーナーを表示)
22 IplImage *dst = cvCreateImage(cvGetSize(src), IPL_DEPTH_8U,3);
23
24 /* グレースケール化 */
25 cvConvertImage(src,gray,NULL);
26 /* ノイズ除去のためのフィルタ */
27 cvSmooth(gray,gray,CV_GAUSSIAN,3,3);
28 /* 結果表示用にコピー */
29 cvCopy(src,dst);
30
31 /* コーナー検出 */
32 cvGoodFeaturesToTrack(gray,eigen,temp,corners,&count,0.01,5,NULL,3,1,0.05);
33
34 /* 処理結果を書き込む（検出したコーナーを円で囲む）*/
35 for(int i=0; i<count; i++)
36 {
37 CvPoint pt = cvPointFrom32f(corners[i]);
38 cvCircle(dst,pt,2,CV_RGB(255,0,0),-1,CV_AA,0);
39 }
40
41 cvNamedWindow("Corner Detection",1);
42 cvShowImage("Corner Detection",dst);
43
44 cvWaitKey(0);
45 cvDestroyWindow("Corner Detection");
46
47 /* メモリ解放 */
48 cvReleaseImage(&src);
49 cvReleaseImage(&gray);
50 cvReleaseImage(&eigen);
51 cvReleaseImage(&temp);
52 cvReleaseImage(&dst);
53
54 return 0;
55 }

```

3.2 解説

続いて、ソースコードの解説を行う。1行目～3行目はopenCVのヘッダファイルをインクルード。OpenCVを使用する際は必ずインクルードすること。8行目のCvPoint2D32fであるが、これはOpenCV独自の型であり、浮動小数点型（単精度）座標系の2次元の点を格納する構造体である。内部構造は次の通り。

```

typedef struct CvPoint2D32f
{
    float x;//X座標、通常は0が原点
    float y;//Y座標、通常は0が原点

```

```

}CvPoint2D32f;

コンストラクタ
inline CvPoint2D32f cvPoint2D32f(double x , double y);

CvPoint からの変換
inline CvPoint2D32f cvPointTo32f(double x , double y);

```

12 行目～22 行目で画像の読み込み等、メモリの確保を行う。

Iplimage は openCV の上位ライブラリである "Intel Image Processing Library" の IPL 画像ヘッダ構造体であり、openCV では Iplimage フォーマットの一部のみサポートしている。なお、Iplimage の説明は今回は割愛する。

本プログラムでは入力画像を格納する src、グレースケール画像を格納する gray、作業用の一時画像を保存する eigen および temp、結果を表示する dst の 5 つを宣言しており、それぞれのメモリサイズはそれぞれ順に 8 ビット、8 ビット、32 ビット、32 ビット、8 ビットである。

14 行目の cvLoadImage 関数を用いて、図に挙げる入力画像の読み込みを行っている。



図 2: 入力画像

gray、eigen、temp、dst の作成に用いている cvCreateImage 関数はヘッダの作成とデータ領域の確保を行う関数であり、一つ目の引数で入力画像の高さと幅を取得、2 つ目の引数で画像要素(ピクセル)のビットデプスの指定 IPL_DEPTH_8U は符号なし 8 ビット整数を IPL_DEPTH_32F は単精度浮動小数点数を表している。3 つ目の引数は要素(ピクセル)毎のチャンネル数を表しており、1～4 のどれかを指定する。

25 行目で入力画像 src を 16 行目でメモリ確保したグレースケール領域 gray に cvConvertImage 関数を用いてコンバートし、グレースケール化を行っている。



図 3: グレースケール化画像

以降、本プログラムではこのグレースケール化された画像に対して、各種処理を行っていく。

27 行目で cvSmooth 関数を用いて、画像のノイズを除去するためにガウシアンフィルタを用いて、全てのピクセルに対して 3×3 ピクセルの範囲で平滑化を行う。

29 行目で入力画像上に抽出した特徴点を示すために結果画像領域に入力画像をコピーする。
 32 行目で cvGoodFeaturesToTrack 関数を用いて、画像上の特徴点を抽出し、その座標を第 4 引数の corners に格納していく。35 ~ 39 行目の for 文内で結果画像に抽出した特徴点を描画していく。先ず、37 行目で cvGoodFeaturesToTrack 関数で得た特徴点の座標を画像上の座標に変換する。続いて、37 行目で得られた座標を中心として半径 2 ピクセルの円を描画し、赤色で塗りつぶしていく。



図 4: 結果画像

41 行目で結果画像を PC 画面上に表示するためのウィンドウを作成し、42 行目で結果画像を PC 画面上に表示している。44 行目でキー入力待ちを行い、キーが押されたら 45 行目でウィンドウを PC 画面上から消去する。最後に 48 ~ 52 行目でメモリを解放し、54 行目でプログラムを終了させている。

4 特徴点抽出実験

次に実際に特徴点を抽出する実験について述べる。

最小固有値の閾値を 0.01 ~ 0.1 まで 0.01 刻みで変化させ、各々の固有値の閾値において特徴点間の距離の閾値を 1 ~ 5 まで変化させた場合の特徴点の抽出数を観測する。

5 実験結果

実験した結果は表 1 のようになった。

表 1: 各固有値における抽出された特徴点数

固有値閾値 距離閾値	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
1	530	487	450	411	391	376	364	349	332	320
2	530	487	450	411	391	376	364	349	332	320
3	415	386	363	334	322	310	304	292	285	277
4	346	319	303	282	271	260	256	247	242	234
5	283	267	257	241	230	222	219	210	205	201

どの固有値閾値においても、距離閾値 1 および 2 の場合に同じ数の特徴点を抽出することが確認できた。これは今回の実験に用いた図において、コーナー間の距離がどれも 2 ピクセル以上離れていたためであると考えられる。また、予測通りに固有値の閾値が大きくなればなるほど、コーナー間の距離閾値が大きくなればなるほど抽出できる特徴点の数が減少していくことが確認できた。



(a) コーナー間距離閾値 1

(b) コーナー間距離閾値 2

(c) コーナー間距離閾値 3



(d) コーナー間距離閾値 4

(e) コーナー間距離閾値 5

図 5: 最小固有値閾値 0.01



(a) コーナー間距離閾値 1

(b) コーナー間距離閾値 2

(c) コーナー間距離閾値 3



(d) コーナー間距離閾値 4

(e) コーナー間距離閾値 5

図 6: 最小固有値閾値 0.02



図 7: 最小固有値閾値 0.03



図 8: 最小固有値閾値 0.04



図 9: 最小固有値閾値 0.05



図 10: 最小固有値閾値 0.06



図 11: 最小固有値閾値 0.07



図 12: 最小固有値閾値 0.08



図 13: 最小固有値閾値 0.09



図 14: 最小固有値閾値 0.1

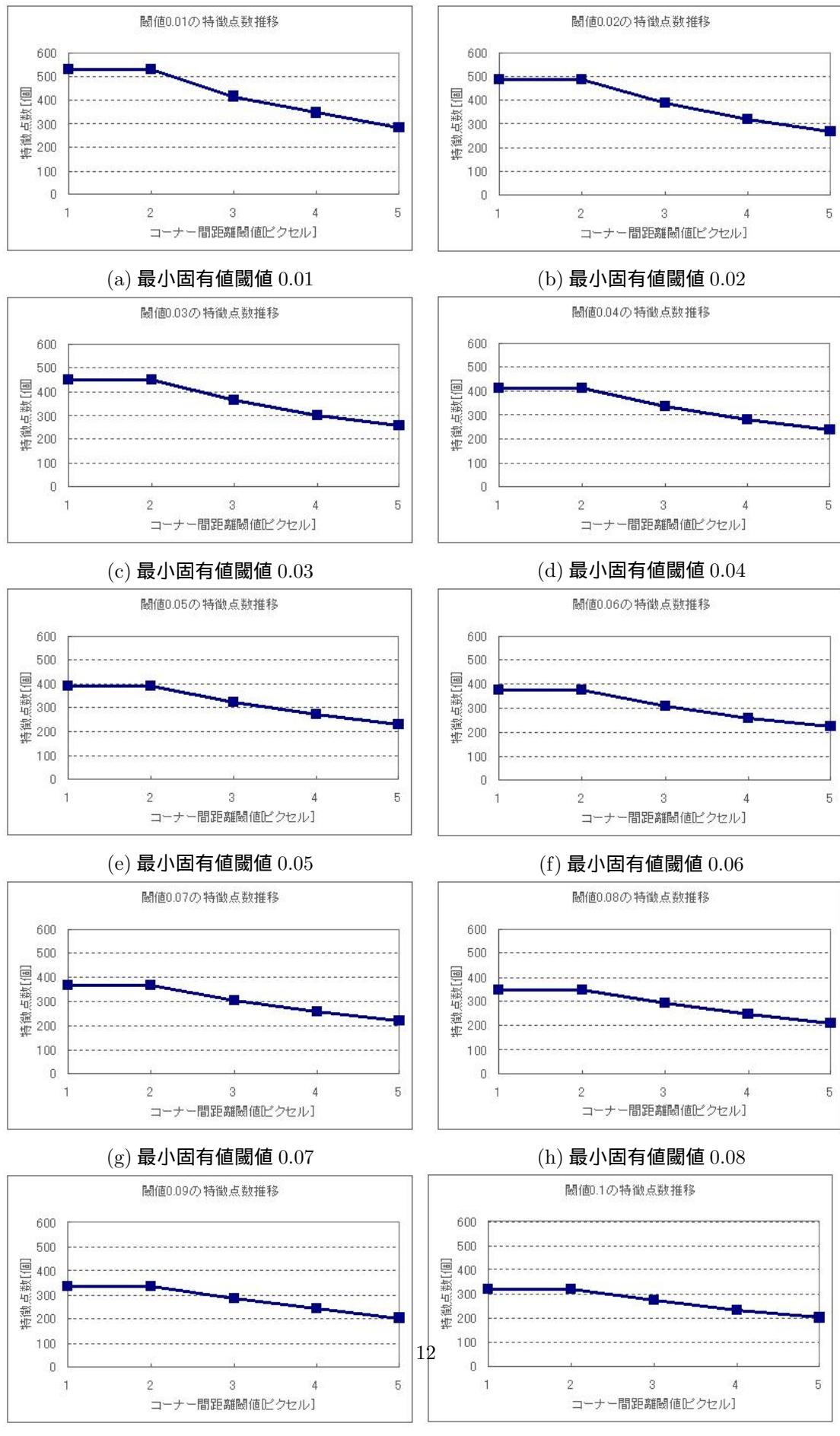


図 15: 実験結果

6 最後に

今回は画像上の特徴点を抽出する方法の一例として，openCV の cvGoodFeaturesToTrack 関数を用いる方法を紹介した．本稿で用いた手法を用いれば，簡単なプログラミングで画像上から特徴点を抽出できるため，非常に有用であることが確認できた．今後は今回の実験で得られたデータを活かして，卒業研究での特徴点を抽出するための各パラメータの決定を行っていく．

7 参考文献

参考文献

- [1] 金澤靖，金谷健一，“解説 コンピュータビジョンのための画像の特徴点抽出”，電子情報通信学会誌，87-12(2004-12)，1043-1048
- [2] openCV リファレンスマニュアル（日本語），<http://opencv.jp/reference/>