

# 1 目的

Python のライブラリについて理解を深め、画像処理技術の第一歩を学ぶとともに、基本となる OpenCV の基礎的な使い方を学ぶ。

## 2 理論

### 2.1 平滑化フィルタ

平滑化フィルタはノイズの低減やぼかしなどを行う、基本的なフィルタ処理の一つである。

あるピクセルを中心として  $3 \times 3$  などの大きさのオペレーターを置く。そしてオペレーター内のピクセル値の平均をそのピクセルの値としてすることで、画像全体がぼやけたような効果が得られる。

ランダムノイズの低減に効果的だが、画像のディテールやエッジも同時にぼやけてしまう性質がある。

### 2.2 中央値フィルタ

中央値フィルタもノイズ除去を行うフィルタ処理の一種だが、平滑化フィルタとは違い塩胡椒ノイズに強いという性質がある。

平均化フィルタと同じようにオペレーターを置き、オペレーター内のピクセル値の中央値を中心のピクセルの値とする。

この手法はエッジや細部の値をあまり変えることなくノイズを除去でき、計算も比較的容易である。しかしオペレーターが大きすぎると細部が失われてしまうことや塩胡椒ノイズ以外に効果が薄いことが欠点である。

### 2.3 特徴検出の仕組み

#### 2.3.1 グレースケール化

画像全体をグレースケール化して画像処理をより行いやすくする。

#### 2.3.2 ノイズ除去

平滑化フィルタや中央値フィルタを使いノイズを除去して、より特徴検出をしやすくする。

#### 2.3.3 二値化とネガポジ反転

しきい値を決めそれを使って画素を白と黒に分ける処理を画像全体に行う。この時ヒストグラムを使うことでしきい値の調整がやりやすくなる。

その後特徴検出のため白と黒の画素値を反転するネガポジ反転を行う。

### 2.3.4 特徴検出

しきい値の調整がうまくいくと連続した塊が表れる。これをプロブという。プロブを囲む円や長方形、プロブの面積や輪郭、形状などを調べることで物体の特徴を検出できる。

## 2.4 エッジ検出

ラプラシアンフィルタやガウシアンフィルタのようにオペレーターの値を調整することで物体のエッジを検出することができる。

## 2.5 ハフ変換

ハフ変換という特徴検出手法がある。ある直線上にある点の座標を  $(x, y)$  としてそれをパラメータ空間  $(\rho, \theta)$  に映す。その点を通る直線に対し  $\rho$  を原点からの距離、 $\theta$  を X 軸と距離がなす角度としたとき式??と表せる。これをヘッセの標準形という。

$$\rho = x \cos \theta + y \sin \theta \quad (1)$$

この式を満たす  $(\rho, \theta)$  は無数に存在するのでパラメータ空間では正弦波が表れる。この処理をエッジの点全体にかけることで重なる点が出てくる。その中で投票をし一番得票数の多かった点  $(\rho, \theta)$  をある直線の距離と角度として検出する。

円のときも円上の点  $(x, y)$  に対して同じ処理をすることで円を検出できる。このとき円の中心座標  $(p, q)$  と半径  $r$  で式??と表せる。

$$r^2 = (x - p)^2 + (y - q)^2 \quad (2)$$

## 3 方法

### 3.1 物体検出

画像に平均化フィルタと中央値フィルタの両方をかけたあと、画素値のヒストグラムを作りそれを参考に二値化を施しネガポジ反転する。そして BoundingRect 関数と rectangle 関数を使って長方形を描く。

### 3.2 円検出

OpenCV ライブラリの機能である Canny 関数と HoughCircles 関数を使ってハフ変換を行う。そこから中心点の座標と半径を得て Circle 関数を使い円を描く。

## 4 結果

### 4.1 物体検出

結果は図??～??のようになった。ソースコードはコード??



図 1 元画像



図 2 平滑化フィルタ後



図 3 中央値フィルタ後

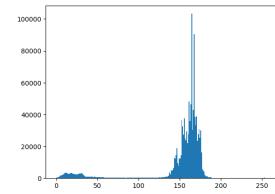


図 4 ピクセル値のヒストグラム



図 5 二値化後

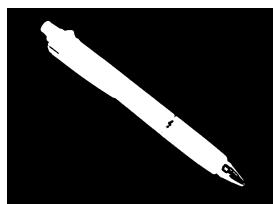


図 6 ネガポジ反転



図 7 結果画像

ソースコード 1 物体検出のソースコード

```
1 import cv2
2
3 import numpy as np
4
5 from matplotlib import pyplot as plt
6
7 img_org = cv2.imread('Square_org.png',cv2.IMREAD_GRAYSCALE)
8
9 #画像をグレースケール化して読み込み
10
11 img_ave = cv2.blur(img_org,(3,3)) #平滑化フィルタ処理
12
13 img_med = cv2.medianBlur(img_ave,3) #中央値フィルタ処理
14
15 plt.hist(img_med.ravel(),256,[0,256]) #ヒストグラムにプロット
16 plt.show() #プロットを表示
17
18 th = 130 #しきい値設定
19
20 ret,img_dst = cv2.threshold(img_med,th,255,cv2.THRESH_BINARY) #二値化処理
21
22 img_dst2 = 255 - img_dst #ネガポジ反転
23
24 x,y,w,h = cv2.boundingRect(img_dst2) #特徴検出
25
26 img = cv2.rectangle(img_org,(x,y),(x+w,y+h),(0,0,0),2) #長方形を描画
27
28 cv2_imshow(img) #結果画像を表示
```

## 4.2 円検出

結果は図??～??のようになった。ソースコードはコード??



図 8 元画像



図 9 グレースケール化  
後

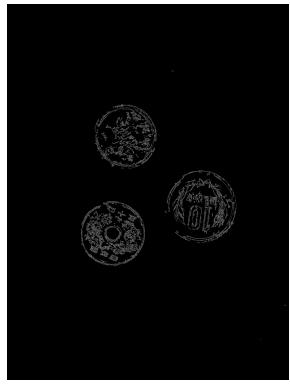


図 10 エッジ検出の可視化



図 11 結果画像

### ソースコード 2 円検出

```
1 import cv2
2 import numpy as np
3
4 img_org = cv2.imread('Circle_org',cv2.IMREAD_GRAYSCALE)
5 img_edge = cv2.Canny(img_org,10,250)
6 img_med = cv2.medianBlur(img_org,3)
7 circles = cv2.HoughCircles(image=img_med,method=cv2.HOUGH_GRADIENT,
8                             dp=2,minDist=100,param1=200,param2=400,
9                             minRadius=70,maxRadius=400)
10 circles = circles.astype('int')
11 img_out = img_org.copy()
12 for p,q,r in circles[0]:
13     cv2.circle(img_out,(p,q),r,(0,255,0),2)
14     cv2.circle(img_out,(p,q),5,(255,0,0),5)
15 cv2.imshow(img_out)
```

## 5 考察

### 5.1 物体検出

#### 5.1.1 実験 1

$3 \times 3$  から  $7 \times 7$  にしたらよりピクセル値が均一になった。ピクセル値がすべて 3 枠の数字になった。

#### 5.1.2 実験 2

しきい値が 150 だとノイズが入った。

図??より 130 くらいから増えていたのでしきい値を 130 にした。

長方形がしっかりとボールペンの先端とキャップ側を囲えていた。

#### 5.1.3 実験 3

結果は表??のようになった。

表 1 実験 3 の結果

		(第 1 引数, 第 2 引数)			
		(低, 低)	(低, 高)	(高, 高)	(高, 低)
第 1 引数と第 2 引数の差	近い	ノイズ多い。変な箇所あり。	×	ノイズ少ない。輪郭が変。	×
	遠い	×	ノイズ少ない。輪郭はしっかりとしている。	×	ノイズ少ない。輪郭はしっかりとしている。

### 5.2 実験 4

最初 50 円玉の中心だけが複数個検出された。なので minDist と minRadius、maxRadius を上げた。

すると変な検出がいくつか出てきた。なので param1=100、param2=200、maxRadius=400 築上げた。

するとコインの外側だけを検出することができた。

## 6 課題

### 6.1 フィルタ処理について、他にどんな物があるか調査せよ [highpath]

#### 6.1.1 ハイパスフィルタ

高周波数成分は残し、低周波数成分は除去するようなフィルタをハイパスフィルタという。周波数成分とは画像の急激な変化のことである。低周波数を除去し高周波数は残すので、画像のエッジや細かい模様を強調することができる。

### 6.2 物体や園の特徴の検出以外に、他にどんなものがあるか調査せよ

#### 6.2.1 傾き補正

画像の文字や文字行の傾きを最小外接矩形を使うと OpenCV のみで傾きの補正が行える。

グレースケール、ノイズ処理、二値化処理をする。そして文字や文字行を構成するすべての輪郭点を findContours 関数で集める。集めた点に対して minAreaRect 関数を適用し、矩形の回転角度を取得する。取得した角度を補正角として getRotationMatrix2D 関数と warpAffine 関数により回転し、水平化する。

これを行うことで画像の文字や文字行の傾きを補正できる。