

カイ二乗値の最小化

2020.01.15 谷

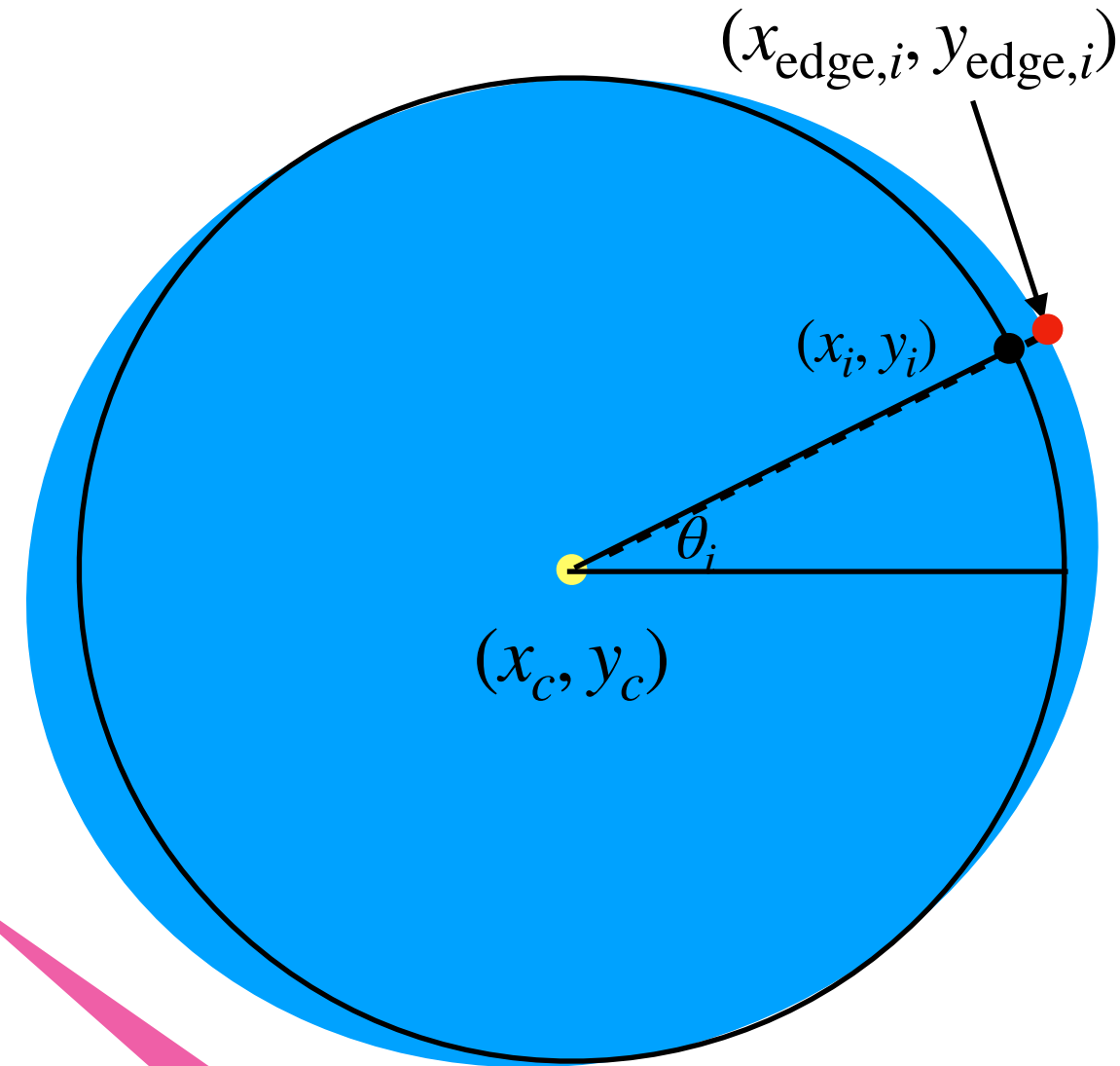
カイ二乗値の最小化

$$E_{\text{sum}}(x_c, y_c, r) = \sum_i \left(|x_i - x_{\text{edge},i}|^2 + |y_i - y_{\text{edge},i}|^2 \right)$$

$$x_i = x_c + r \cos \theta_i, \quad y_i = y_c - r \sin \theta_i$$

E_{sum} が最小となる (x_c, y_c, r) を求める。

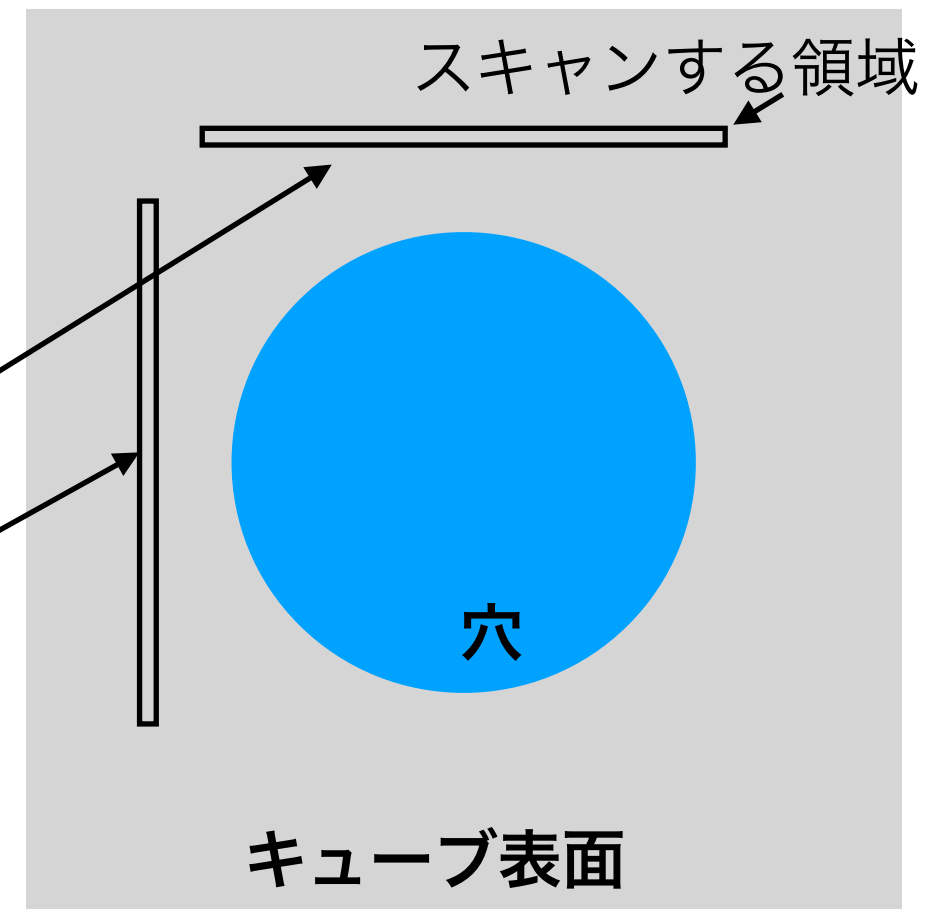
- (x_c, y_c) は円の中心、 r は円の半径
- $x_{\text{edge}}, y_{\text{edge}}$ は穴のエッジ上の点
- (x_c, y_c, r) の初期値は以前やった、二値化画像から得た中央値を使う。
- 今回はエッジ上の16点について和をとった $(\theta_i = 0, \frac{\pi}{8}, \frac{\pi}{4}, \dots, \frac{15}{8}\pi)$ 。



二値化の threshold は
各画像ごとに決定

画像の二値化

- 二値化のための threshold の決め方
- 穴の周囲の色の平均を参照値とする。
- 参照値の候補をふたつ（orそれ以上）用意する
 - x方向にスキャンした平均
 - y方向にスキャンした平均
- キューブ表面の傷・印等の影響を減らすため、候補のうち、最も白いものを参照値として採用する。



今回は以前提案していただいたように、参照値の 30% カットの値を threshold として用いた

最小化の手順

ある $\mathbf{x} = (x, y, r)$ での Esum の勾配

$$\nabla E = \left(\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}, \frac{\partial E}{\partial r} \right)$$

を求め、設定した learning rate η をかけて (x, y, r) から引き更新する。

$$\mathbf{x}' = \mathbf{x} - \eta \nabla E$$

この操作を指定ステップ数だけ繰り返す。

徐々に最小化されている様子がわかる
(右の出力参照)。

時間もそれほどかかっていない。

learning_rate = 0.01, steps = 50
initial_position = [356.099998474, 244.099998474, 51.4]
はじめのEsumの値:294.19271881806037

gradient[0]=[3.19995117 17.19995117 -127.88263133]
Esum[0]=154.2483374368868

gradient[1]=[2.1759668 11.6959668 -86.9601893]
Esum[1]=89.53805548782361

gradient[2]=[1.47965742 7.95325742 -59.13292872]
Esum[2]=59.61602111522834

gradient[3]=[1.00616705 5.40821505 -40.21039153]
Esum[3]=45.78007242172495

gradient[4]=[0.68419359 3.67758623 -27.34306624]
Esum[4]=39.38232974596808

gradient[5]=[0.46525164 2.50075864 -18.59328504]
Esum[5]=36.42401353280427

·
·
·

gradient[47]=[4.295230e-08 2.309263e-07 -1.716955e-06]
Esum[47]=33.87950940915731

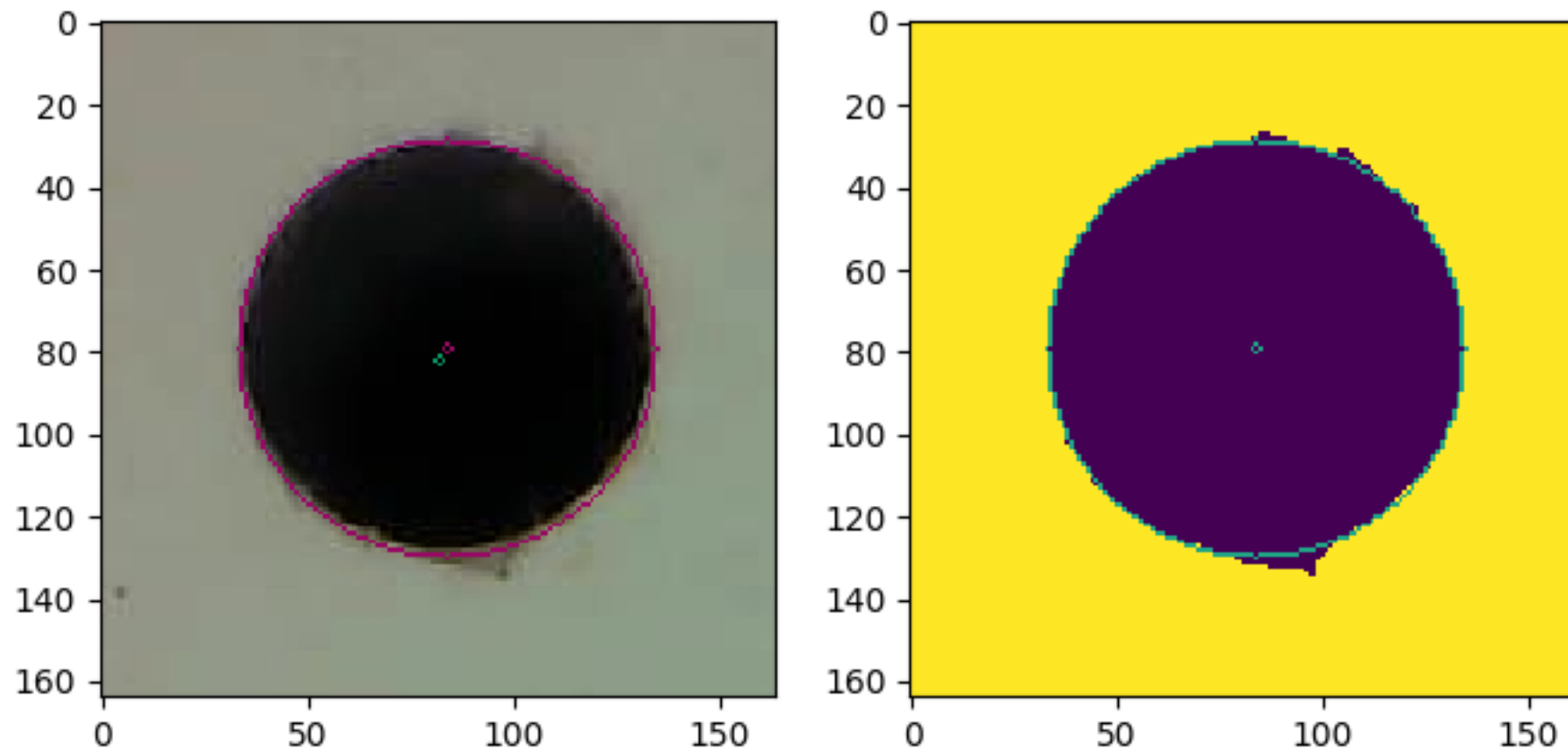
gradient[48]=[2.923883e-08 1.570299e-07 -1.167386e-06]
Esum[48]=33.879509409157265

gradient[49]=[1.982414e-08 1.067590e-07 -7.938183e-07]
Esum[49]=33.87950940915729

勾配法が見つけた最小値でのx: [356. 243.5625 55.396332]
time:0.057205915451049805 sec
Esumの最小値:33.87950940915729

実行例

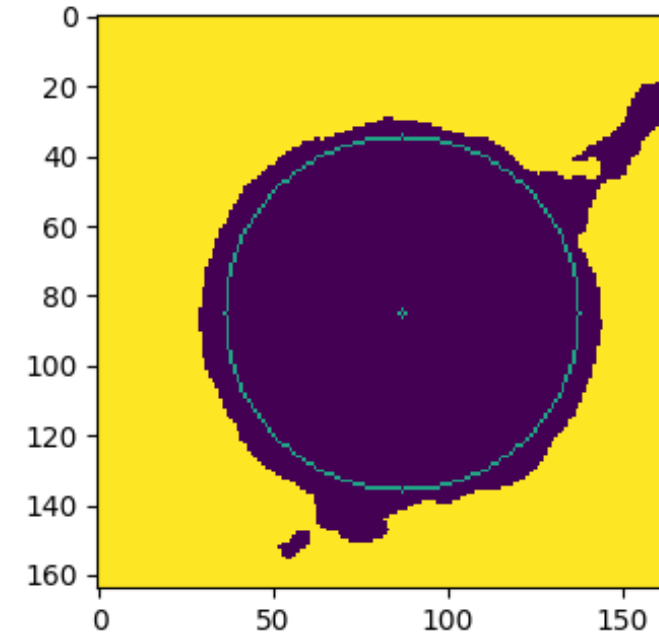
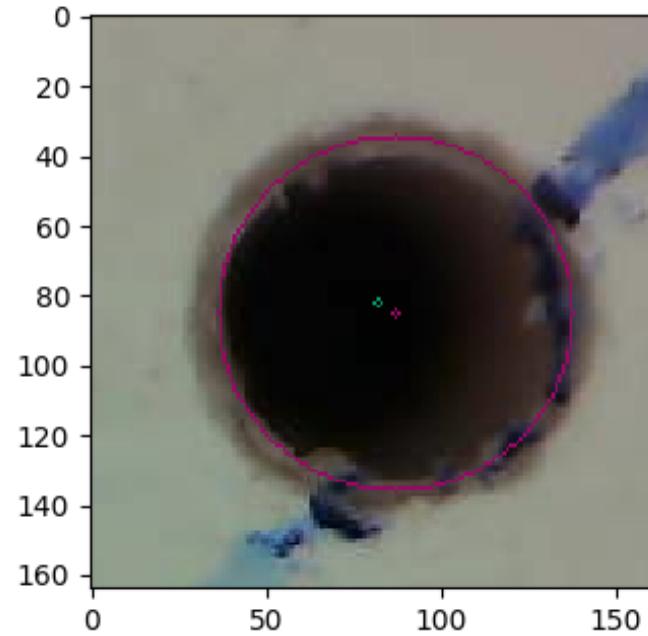
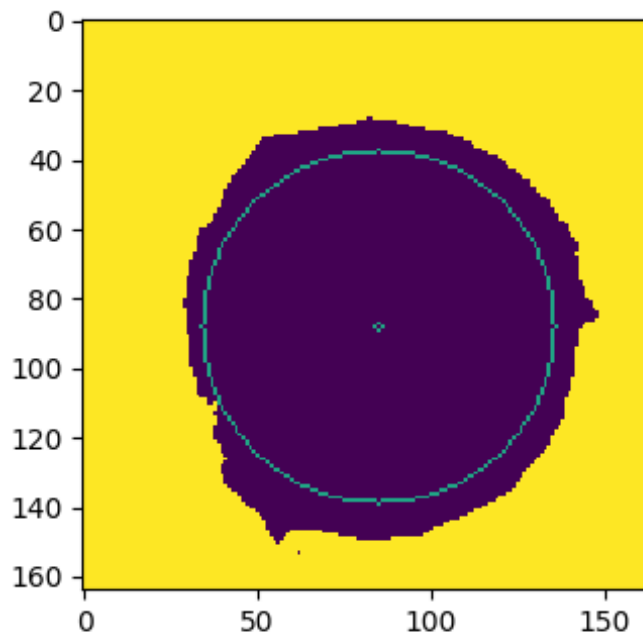
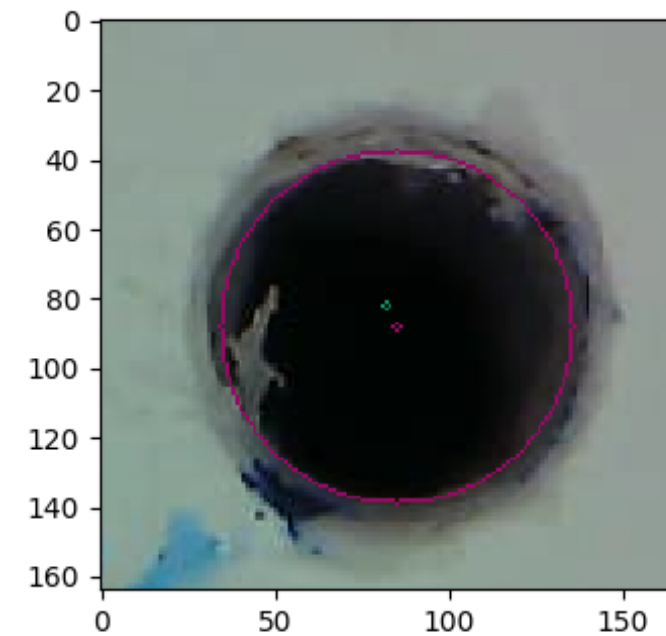
good の例：円は初期 (x, y, r)によるもの



```
learning_rate = 0.01, steps = 50  
initial_position = [364.0999984741211, 230.0999984741211, 52.4]  
はじめのEsumの値:50.57412827491975  
勾配法が見つけた最小値でのx: [363.8125      229.375      51.08272056]  
time:0.04395008087158203 sec  
Esumの最小値:13.078075424774163
```

(x, y, r):(364.1, 230.1, 52.4)→(363.8, 229.4, 51.1)
Esum : 50.5 → 13.1

bad の例：円は初期 (x, y, r)によるもの



learning_rate = 0.01, steps = 50
initial_position = [652.099998474, 254.099998474, 52.4]
はじめのEsumの値:626.501403830376

勾配法が見つけた最小値でのx: [652.0625 253.8125 58.09540]

time:0.04877614974975586 sec
Esumの最小値:106.15383203261482

(x, y, r):(652.1, 254.1, 52.4)
→(652.1, 253.8, 58.1)

Esum : 626.5 → 106.1

learning_rate = 0.01, steps = 50
initial_position = [659.09999847, 217.09999847, 52.4]
はじめのEsumの値:651.4675248444642

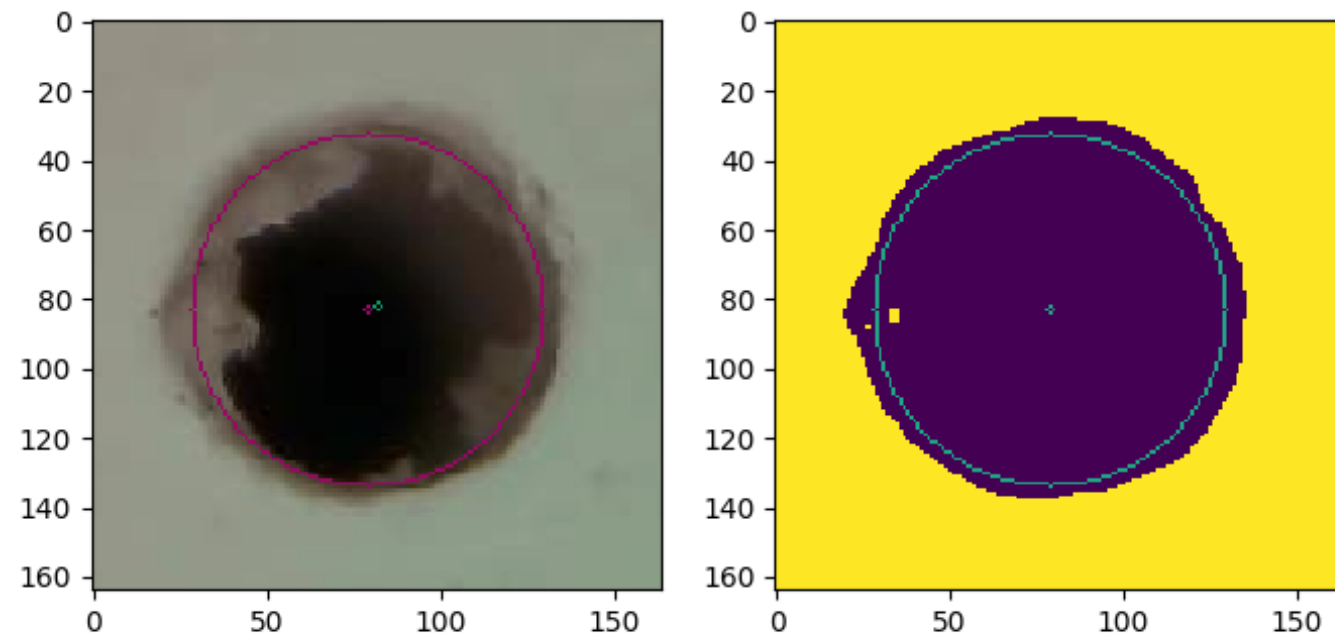
勾配法が見つけた最小値: [658.1875 217.625 57.347693]

time:0.044788360595703125 sec
最小値でのEsumの値:242.06030867727364

(x, y, r):(659.1, 217.1, 52.4)
→(658.2, 217.6, 57.3)

Esum : 651.5 → 242.1
(右上の印に引っ張られている?)

bad の例：円は初期 (x, y, r) によるもの



learning_rate = 0.01, steps = 50
initial_position = [356.09999847, 244.09999847, 52.4]
はじめのEsumの値:182.3100874940509

勾配法が見つけた最小値: [356. 243.5625 55.396332]

time:0.04555487632751465 sec
最小値でのEsumの値:33.87950940915727

$(x, y, r): (356.1, 244.1, 52.4)$

$\rightarrow (356.0, 243.6, 55.4)$

Esum : 182.3 \rightarrow 33.9

- どれも中心 (x, y) については最小化してもほぼ変わらず \rightarrow 中央値作戦はうまくいっている
- r の変化は大きい
- 二値化の threshold によって期待される r の値は変わる。
- 異なる threshold でこの操作を行い、Esum の最小値が最も小さくなるものをその画像での threshold とする？（未実行）