濃淡画像処理:鮮鋭化

画像の鮮鋭化

- 物体の輪郭などがぼやけた画像に対しては，濃度値の変化を強調することで鮮明な画像が得られる

（a） 原画像

（b） 鮮鋭化フィルタによる処理画像
（8近傍鮮鋭化フィルタ）

図 5.12 画像の鮮鋭化の例

山田 宏，末松 良一 「画像処理と画像認識 —AI時代の画像処理入門」 コロナ社より

画像の鮮鋭化

- エッジ部分が強調され，鮮明な画像となる

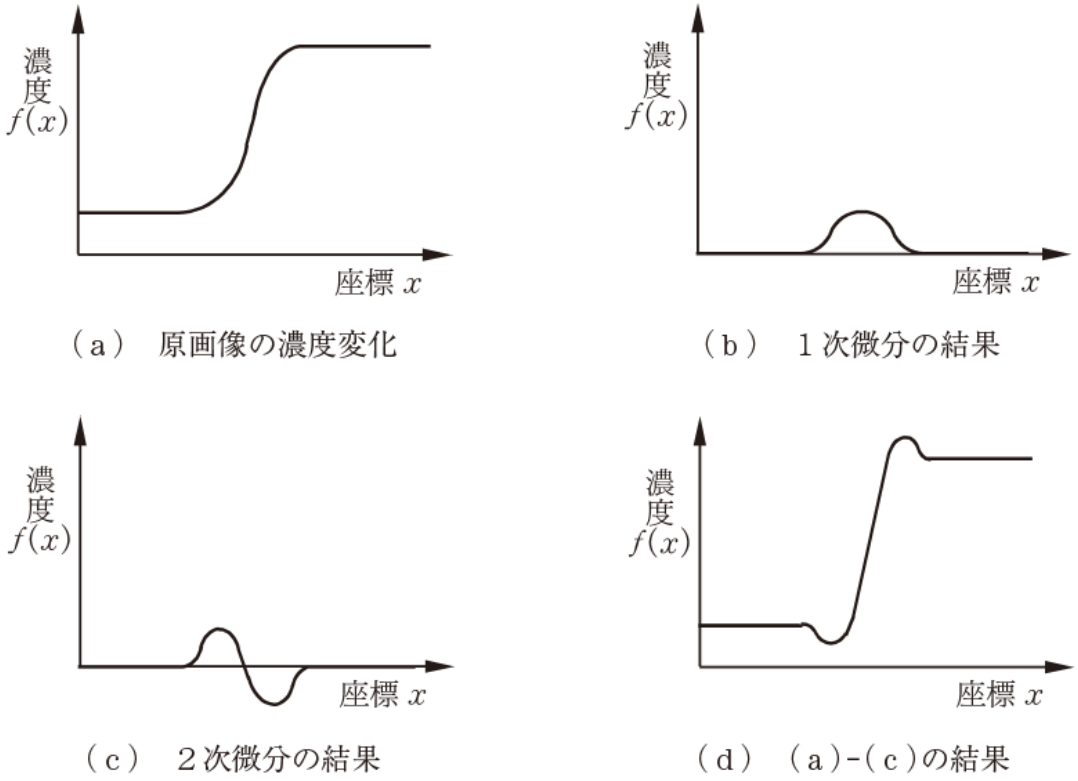


図 5.9 2次微分を用いた鮮鋭化

山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門」 コロナ社より

鮮鋭化フィルタ

- 鮮鋭化フィルタは，原画像から2次微分であるラプラシアンフィルタを引くことで導出できる

0	−1	0
−1	5	−1
0	−1	0

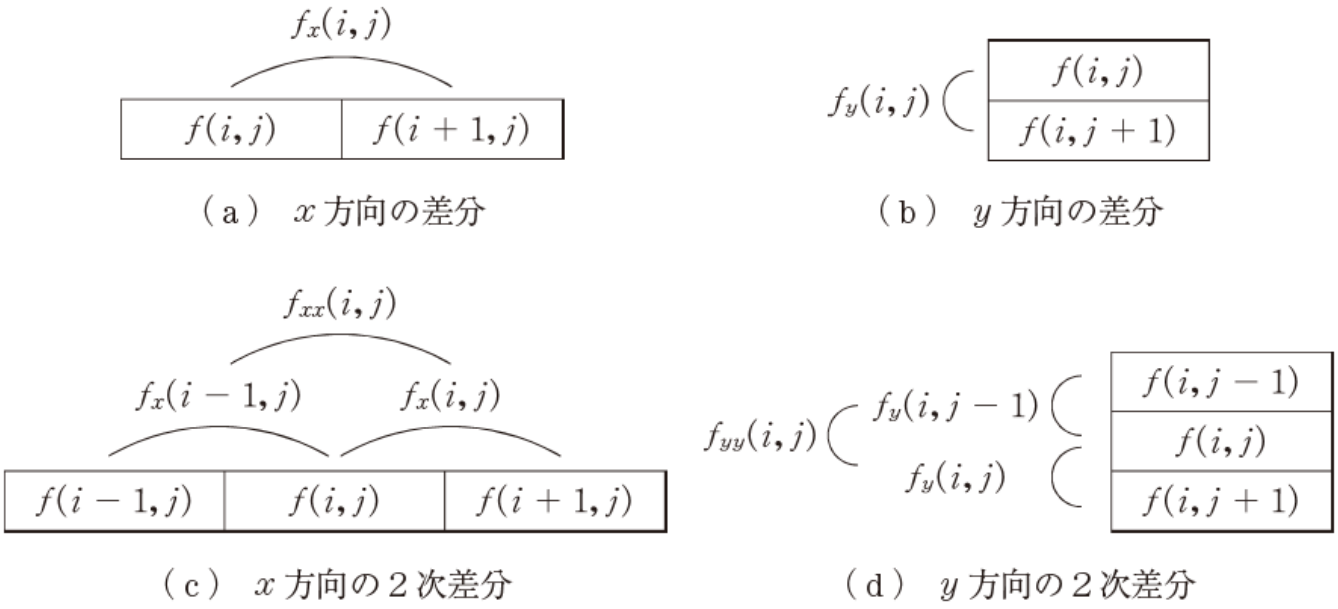
4近傍鮮鋭化フィルタ

−1	−1	−1
−1	9	−1
−1	−1	−1

8近傍鮮鋭化フィルタ

4近傍鮮鋭化フィルタの導出

- ディジタル画像においては，微分は差分により代用されるため，まずは2次微分を計算



（a） x 方向の差分　　（b） y 方向の差分

（c） x 方向の２次差分　　（d） y 方向の２次差分

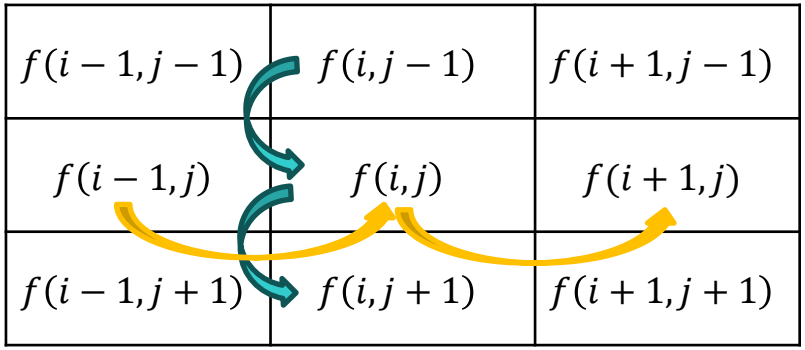
図 5.10　画素の１次微分と２次微分

山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門」 コロナ社より

4近傍鮮鋭化フィルタの導出

- 1次微分

$$f_x(i,j) = f(i+1,j) - f(i,j)$$
$$f_y(i,j) = f(i,j+1) - f(i,j)$$



- 2次微分

0	0	0
1	−2	1
0	0	0

$$\begin{aligned} f_{xx}(i,j) &= f_x(i,j) - f_x(i-1,j) \\ &= \{f(i+1,j) - f(i,j)\} - \{f(i,j) - f(i-1,j)\} \\ &= f(i-1,j) - 2f(i,j) + f(i+1,j) \end{aligned}$$

0	1	0
0	−2	0
0	1	0

$$\begin{aligned} f_{yy}(i,j) &= f_y(i,j) - f_y(i,j-1) \\ &= \{f(i,j+1) - f(i,j)\} - \{f(i,j) - f(i,j-1)\} \\ &= f(i,j-1) - 2f(i,j) + f(i,j+1) \end{aligned}$$

4近傍鮮鋭化フィルタの導出

- ラプラシアン$\nabla^2 f(i,j)$を求める

$$\nabla^2 f(i,j) = f_{xx}(i,j) + f_{yy}(i,j)$$
$$= f(i,j-1) + f(i-1,j) - 4f(i,j) + f(i+1,j) + f(i,j+1)$$

- 元画像との差分$f(i,j) - \nabla^2 f(i,j)$を計算

$$g(i,j) = f(i,j) - \nabla^2 f(i,j)$$
$$= -f(i,j-1) - f(i-1,j) + 5f(i,j) - f(i+1,j) - f(i,j+1)$$

0	0	0
0	1	0
0	0	0

−

0	1	0
1	−4	1
0	1	0

=

0	−1	0
−1	5	−1
0	−1	0

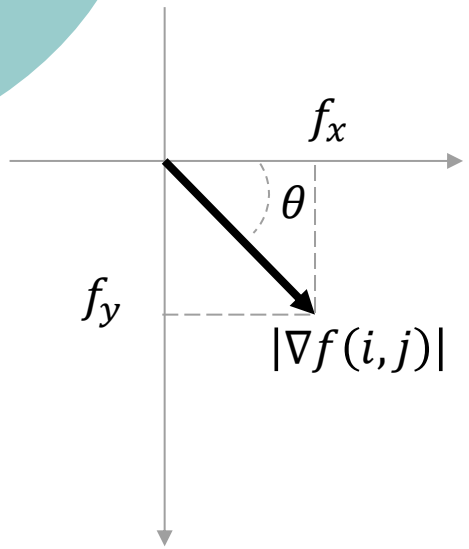
濃淡画像処理:エッジ検出

エッジ検出

- エッジ＝**濃淡が急激に変化**する場所
- 画像の中に表示される**物体の輪郭や線**では一般的に濃淡が急激に変化するため，エッジの検出は重要である
- 様々なエッジ検出手法が提案されている
  - Prewitt/Sobelフィルタ
  - ゼロ交差法
  - Cannyのエッジ検出アルゴリズム

差分によるエッジ検出

- **画素間の差分を取り**エッジの勾配を計算
- **勾配強度**$|\nabla f(i,j)|$と**方向**θを計算してエッジを検出



$$f_x(i,j) = f(i+1,j) - f(i,j)$$

$$f_y(i,j) = f(i,j+1) - f(i,j)$$

$$|\nabla f(i,j)| = \sqrt{f_x^2(i,j) + f_y^2(i,j)}$$

$$\theta = \tan^{-1}\frac{f_y}{f_x}$$

−1	1

x方向の差分
$f_x(i,j)$

−1
1

y方向の差分
$f_y(i,j)$

エッジ強度計算の高速化

- 平方根計算は時間がかかる場合もあるため，
  下記の計算で代用されることもある

$$|\nabla f(i,j)| = \sqrt{f_x^2(i,j) + f_y^2(i,j)}$$

各方向の勾配の**絶対値の和**を計算

$$|\nabla f(i,j)| = |f_x(i,j)| + |f_y(i,j)|$$

各方向の勾配の**絶対値の大きい値**を取る

$$|\nabla f(i,j)| = \max\left(|f_x(i,j)|, |f_y(i,j)|\right)$$

Prewittフィルタ

- 差分では，2つの画素の中間の位置の傾きを求めることになる
- 3×3のフィルタを用いて，**注目画素の差分**を計算

−1	0	1
−1	0	1
−1	0	1

x方向の差分$f_x(i,j)$

−1	−1	−1
0	0	0
1	1	1

y方向の差分$f_y(i,j)$

Sobelフィルタ

- 3×3のフィルタを用いて，注目画素の差分を計算
- Prewittフィルタよりも注目画素に重みをつけて
平滑化をしながら差分を計算

−1	0	1
−2	0	2
−1	0	1

x方向の差分$f_x(i,j)$

−1	−2	−1
0	0	0
1	2	1

y方向の差分$f_y(i,j)$

Sobelフィルタによるエッジ検出

- Sobelフィルタ適用後，強度を計算してエッジ検出

原画像

Sobelフィルタ適用後

ゼロ交差法によるエッジ検出

- 画像の2次微分を行うと，下図のようにエッジの下端と上端で正と負のピークが生じる
- 0となる位置をエッジの中央位置として検出

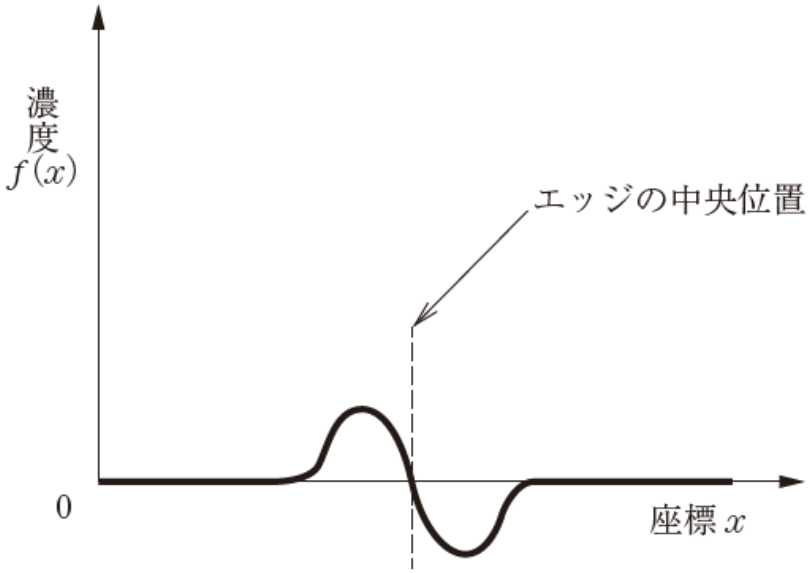


図 5.19　零交差法

山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門」 コロナ社より

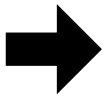
ゼロ交差法によるエッジ検出

- **ガウシアンフィルタ**で平滑化した後，
  ラプラシアンフィルタを適用して2次微分を計算
- ゼロ交差位置を検出

2つのフィルタを1つにまとめる場合もある
LoGフィルタ（Laplacian of Gaussian）
と呼ばれる

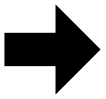
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$
$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

ガウシアンフィルタ



0	1	0
1	−4	1
0	1	0

ラプラシアンフィルタ



$f(i+1,j) * f(i-1,j) < 0$ もしくは
$f(i,j+1) * f(i,j-1) < 0$
あればエッジとして検出

$f(i-1,j-1)$	$f(i,j-1)$	$f(i+1,j-1)$
$f(i-1,j)$	$f(i,j)$	$f(i+1,j)$
$f(i-1,j+1)$	$f(i,j+1)$	$f(i+1,j+1)$

ゼロ交差位置の検出

Cannyのエッジ検出アルゴリズム

- 輪郭線で連続したエッジを検出するための手法として**Cannyのエッジ検出アルゴリズム**が提案されている
- 2つのしきい値を与えてエッジの強度を判定

原画像

輪郭線の検出

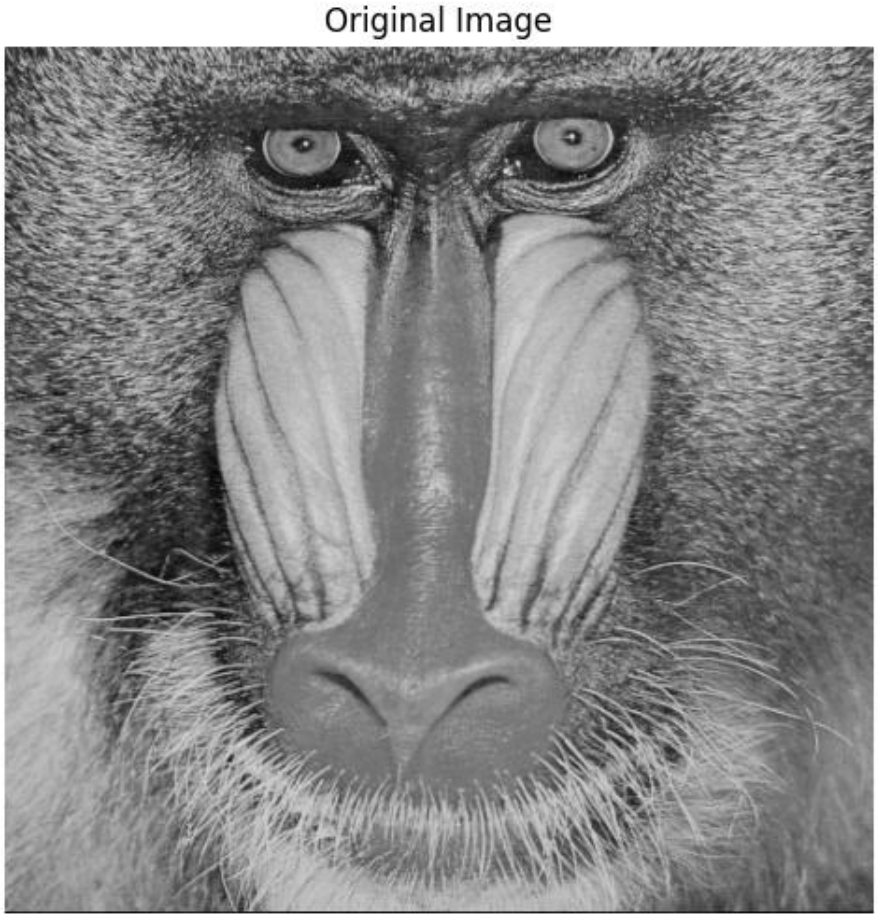
画像処理プログラミング３
平滑化・鮮鋭化・エッジ検出

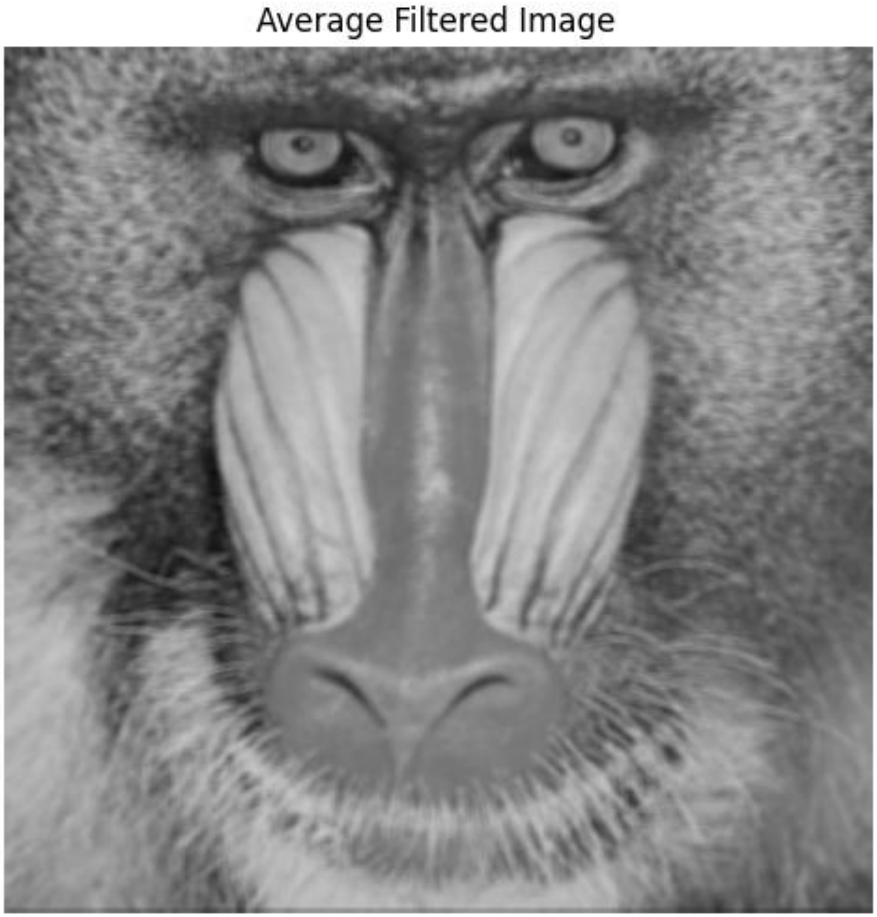
移動平均フィルタによる平滑化

- cv2.blur()で画像に平均値フィルタをかける
- フィルタのサイズに応じてぼやけ方が変化する
  cv2.blur(gray_img, (5, 5)) 赤字部分で指定

```
#画像の読み込み
gray_img = cv2.imread(common_path + 'gray_image.png', cv2.IMREAD_GRAYSCALE)

#移動平均フィルタをかける
average_filtered_img = cv2.blur(gray_img, (5, 5))

# 実行結果の表示
show_images(gray_img,average_filtered_img, 'Original Image', 'Average Filtered Image')
```

移動平均フィルタによる平滑化



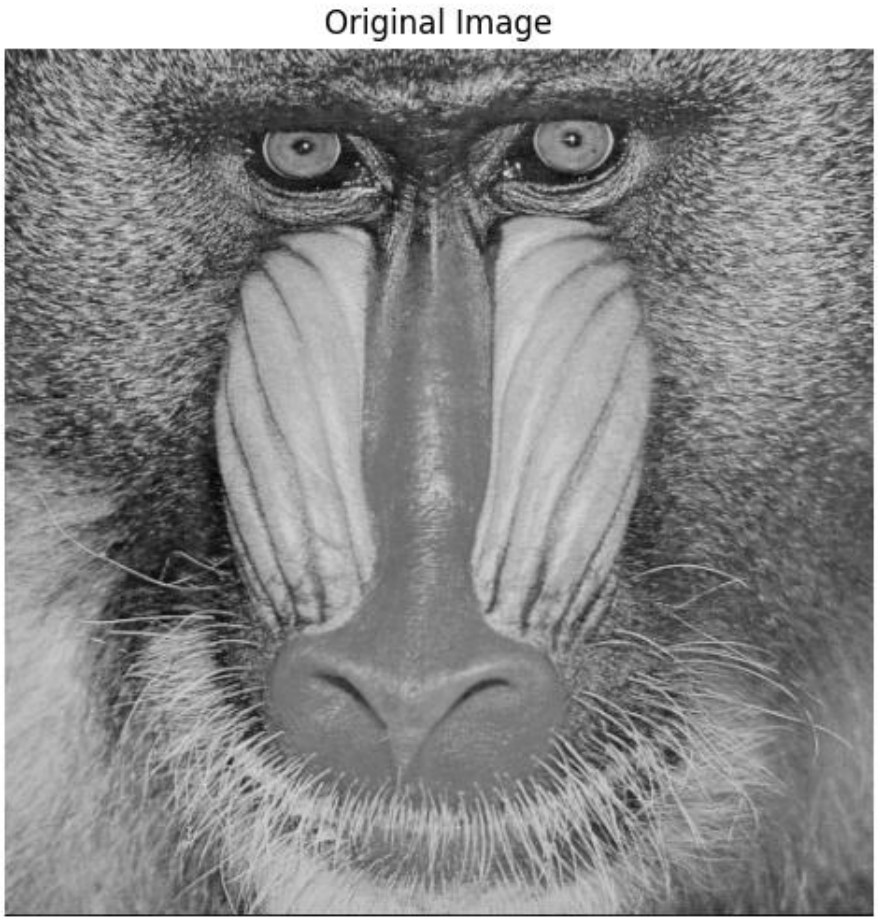


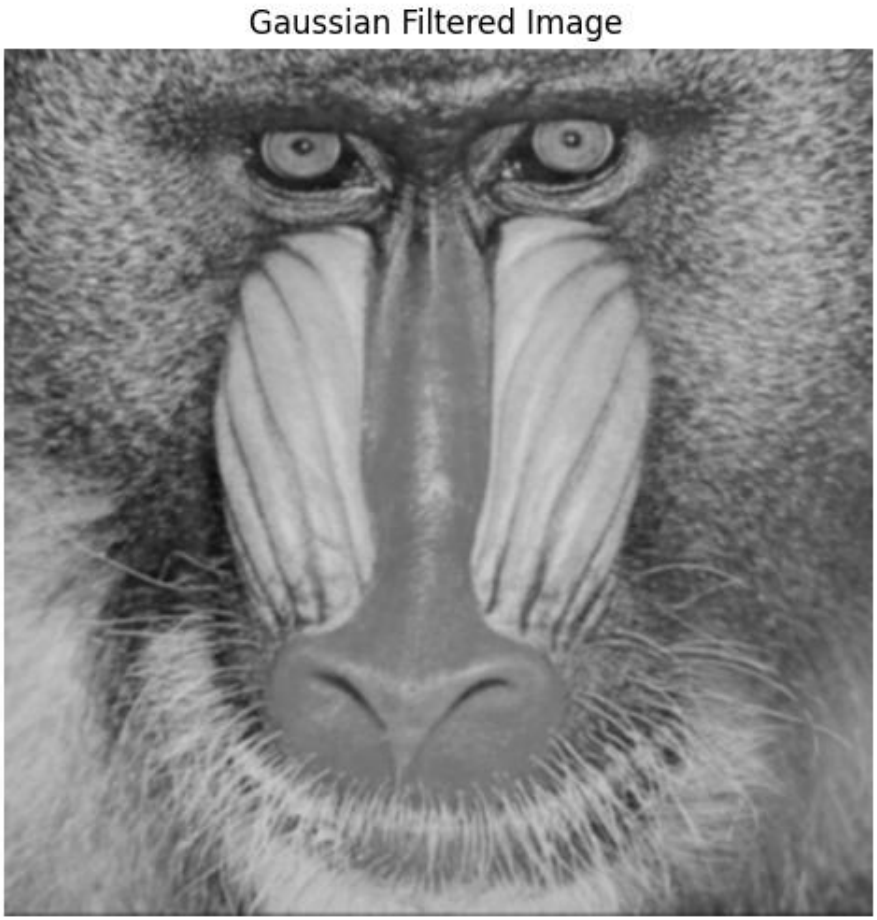
ガウシアンフィルタによる平滑化

- 以後，グレースケール画像の読み込みは省略
- cv2.GaussianBlur(gray_img, (5, 5), 0)
  - 第1引数：入力画像データ
  - 第2引数：フィルタサイズ
  - 第3引数：標準偏差（0にすると自動計算）

```
#ガウシアンフィルタ
gaussian_filtered_img = cv2.GaussianBlur(gray_img, (5, 5), 0)

# 実行結果の表示
show_images(gray_img,gaussian_filtered_img, 'Original Image',
'Gaussian Filtered Image')
```

ガウシアンフィルタによる平滑化



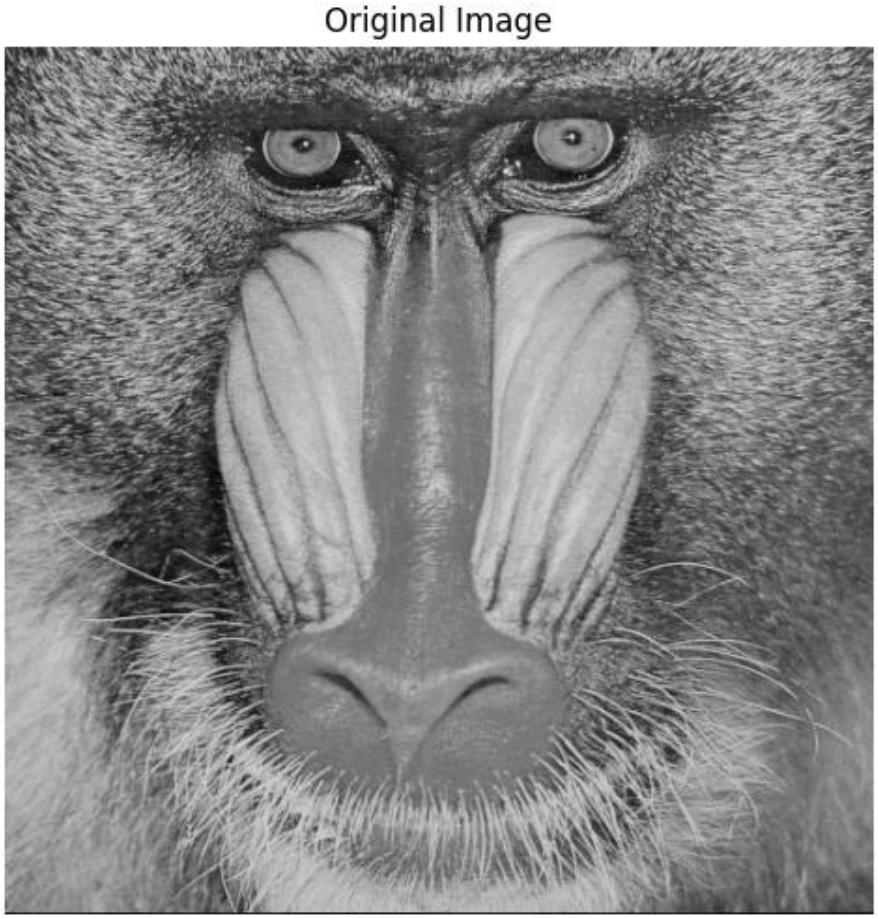


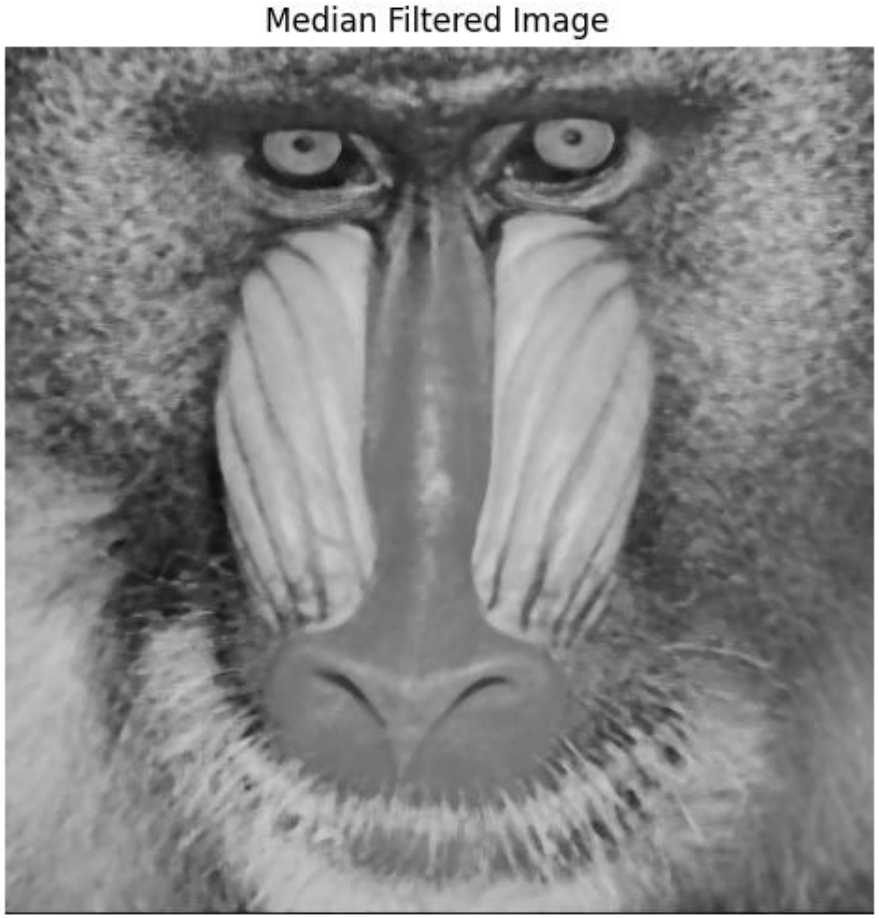
メディアンフィルタによる平滑化

- cv2.medianBlur(gray_img, 5)
  - 第1引数：入力画像データ
  - 第2引数：フィルタサイズ（1次元表記でOK）

```
# メディアンフィルタの適用
median_img = cv2.medianBlur(gray_img, 5)

# 実行結果の表示
show_images(gray_img,median_img, 'Original Image','Median Filtered Image')
```

メディアンフィルタによる平滑化





4近傍鮮鋭化フィルタの適用

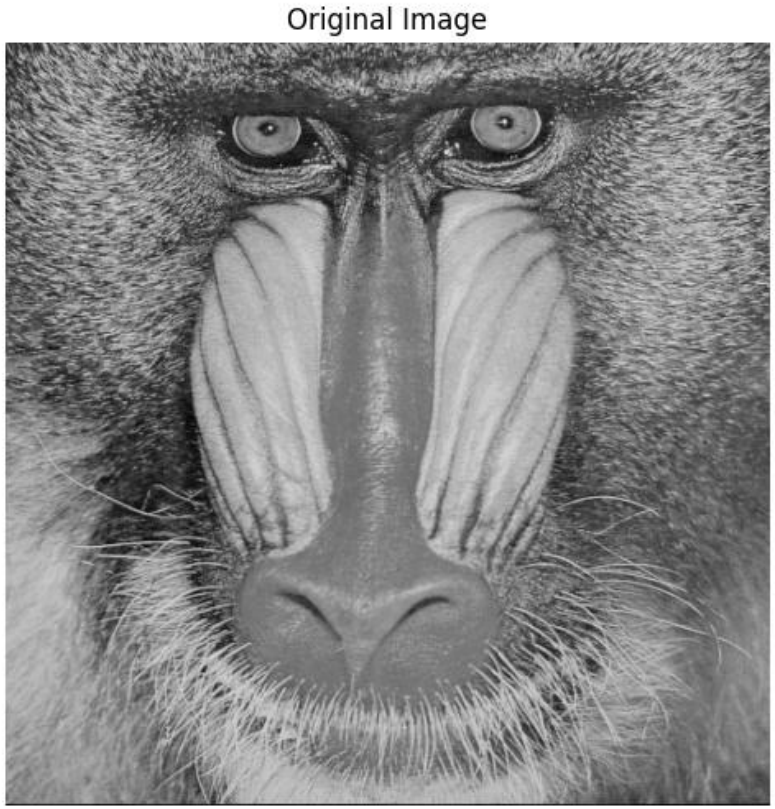
- cv2.filter2D(gray_img, -1, kernel)
  - 第1引数：入力画像データ
  - 第2引数：出力画像のデータ型（今回は-1でOK）
  - 第3引数：入力画像に適用するフィルタ行列

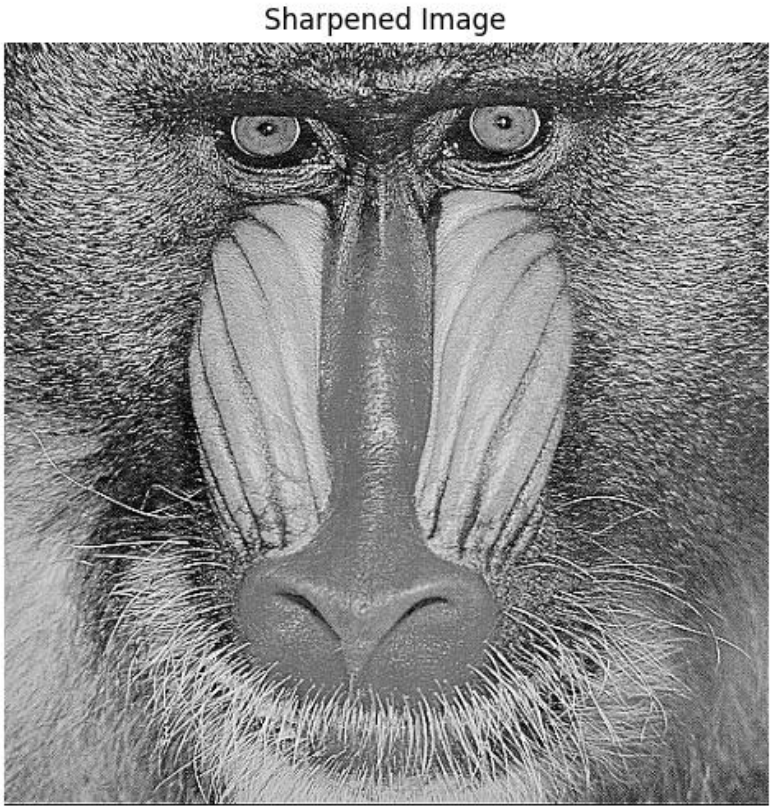
0	−1	0
−1	5	−1
0	−1	0

```
# 4近傍鮮鋭化フィルタの作成
kernel = np.array([[ 0, -1,  0],[-1,  5, -1],[ 0, -1,  0]])

# 鮮鋭化フィルタの適用
sharpened_img = cv2.filter2D(gray_img, -1, kernel)

# 実行結果の表示
show_images(gray_img,sharpened_img, 'Original Image',
'Sharpened Image')
```

4近傍鮮鋭化フィルタ

- ひげなどのエッジが強調され，より鮮明な画像となっている





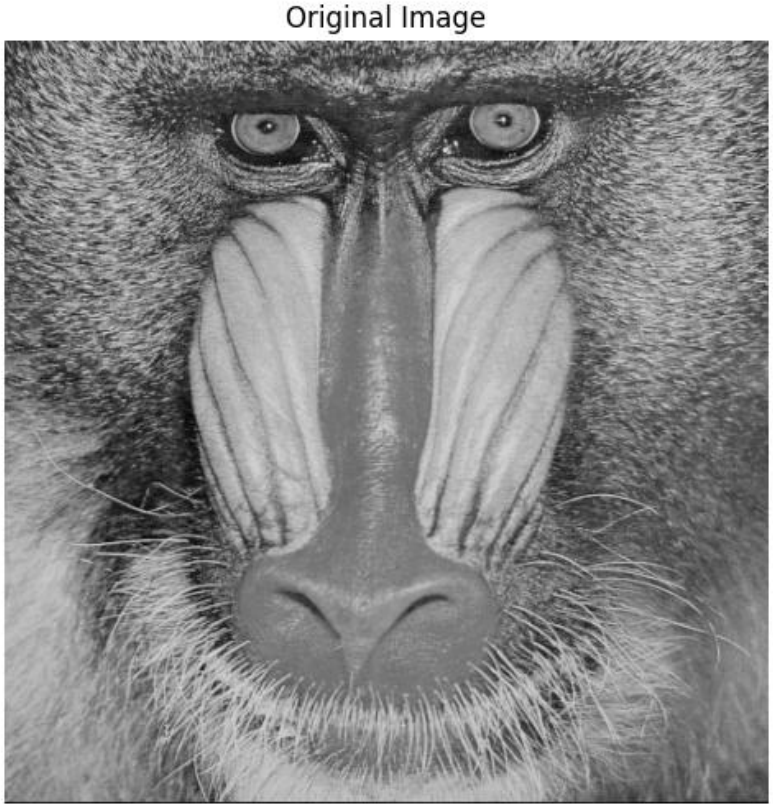
Sobelフィルタによるエッジ検出

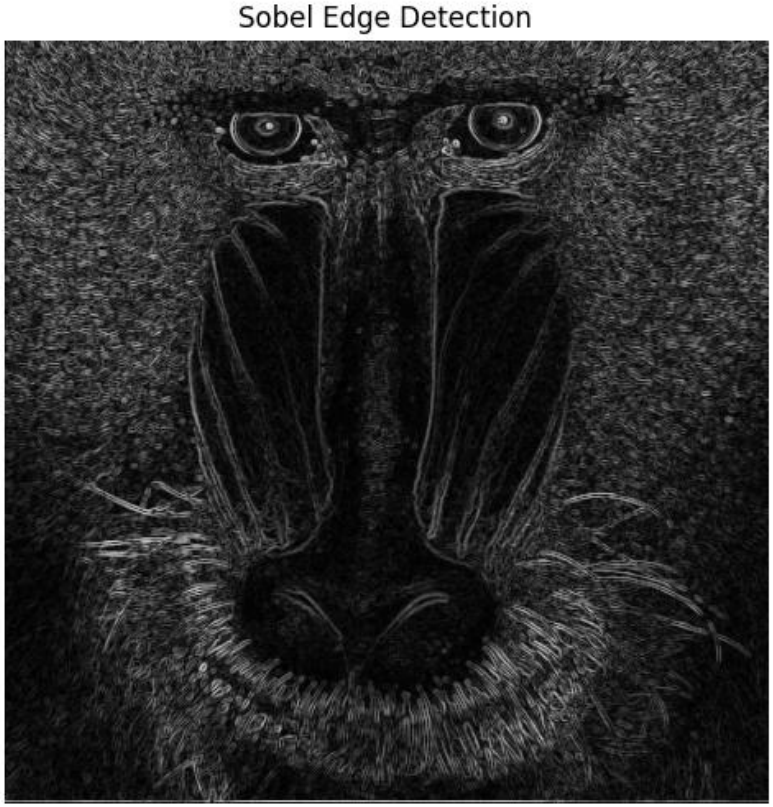
- Sobelフィルタで縦横方向のエッジを検出し強度計算

```
# Sobelエッジ検出
sobel_x = cv2.Sobel(gray_img, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(gray_img, cv2.CV_64F, 0, 1, ksize=3)

#エッジの強度の計算
sobel_edges = cv2.magnitude(sobel_x, sobel_y)

#256階調となるように正規化
sobel_edges_img = cv2.normalize(sobel_edges, None, 0, 255,
cv2.NORM_MINMAX, cv2.CV_8U)

# 実行結果の表示
show_images(gray_img,sobel_edges_img, 'Original Image','Sobel
Edge Detection')
```

Sobelフィルタによるエッジ検出

- 濃度値が急激に変化する箇所がエッジとして検出される



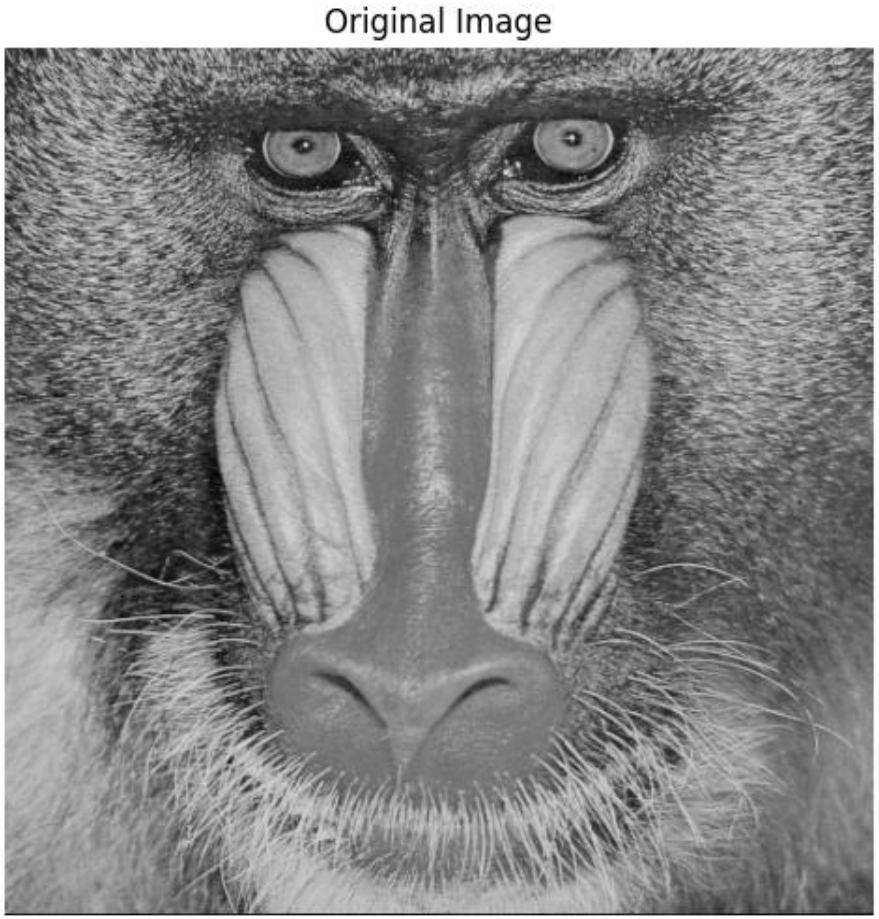


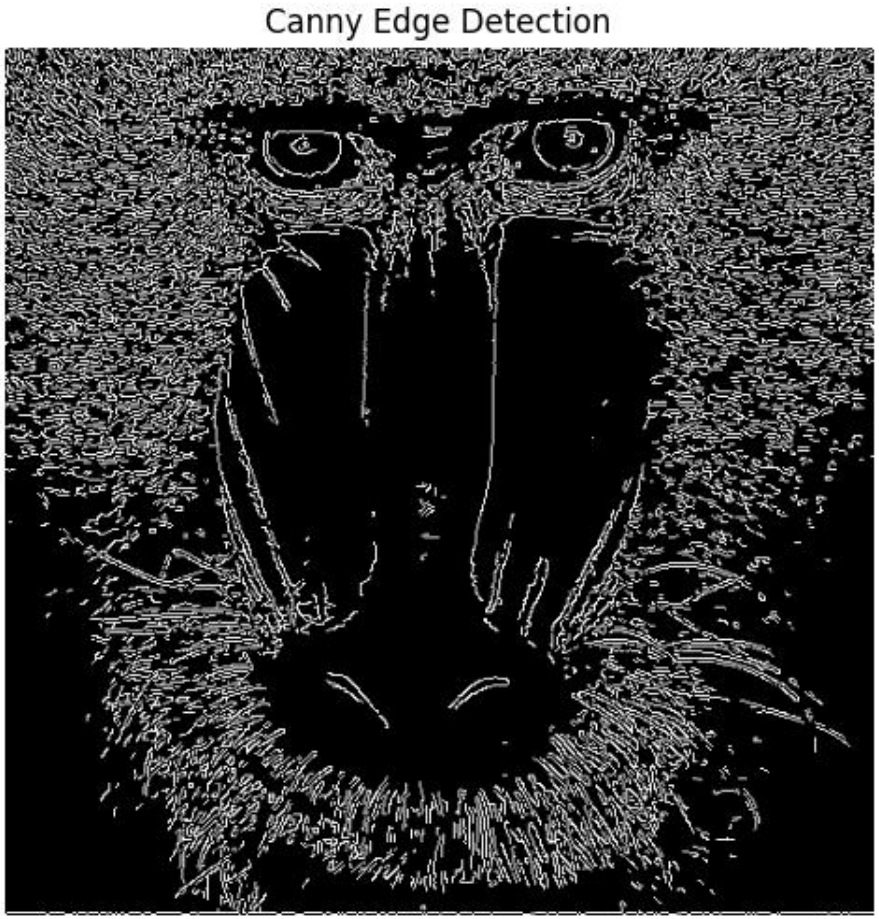
Cannyのエッジ検出アルゴリズム

- Cannyのエッジ検出アルゴリズムを適用
- cv2.Canny(gray_img, 180, 280)
  2つのしきい値により検出できるエッジの強度を制御

```
#Cannyアルゴリズムを使用してエッジ検出を行う
canny_edges_img = cv2.Canny(gray_img, 180, 280)

# 実行結果の表示
show_images(gray_img,canny_edges_img, 'Original Image',
'Canny Edge Detection')
```

Cannyのエッジ検出アルゴリズム





しきい値による結果の違い

- 大きい方のしきい値を固定して，小さい方のしきい値を小さくすると，細かいエッジが検出される

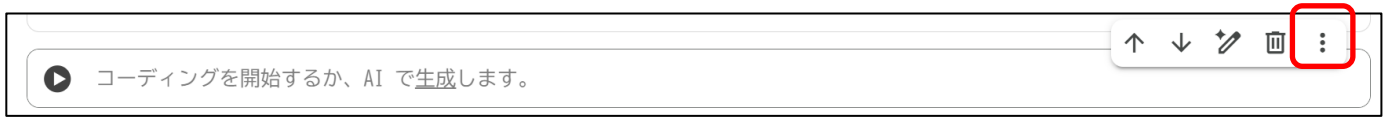
(100,280)

(180,280)

(280,280)

（参考）しきい値を変えられるようにする

- 新たにセルを追加し，
  右端の三点リーダー＞フォームを追加をクリック



- 三点リーダー＞フォーム＞フォームの項目の追加をクリック

（参考）しきい値を変えられるようにする

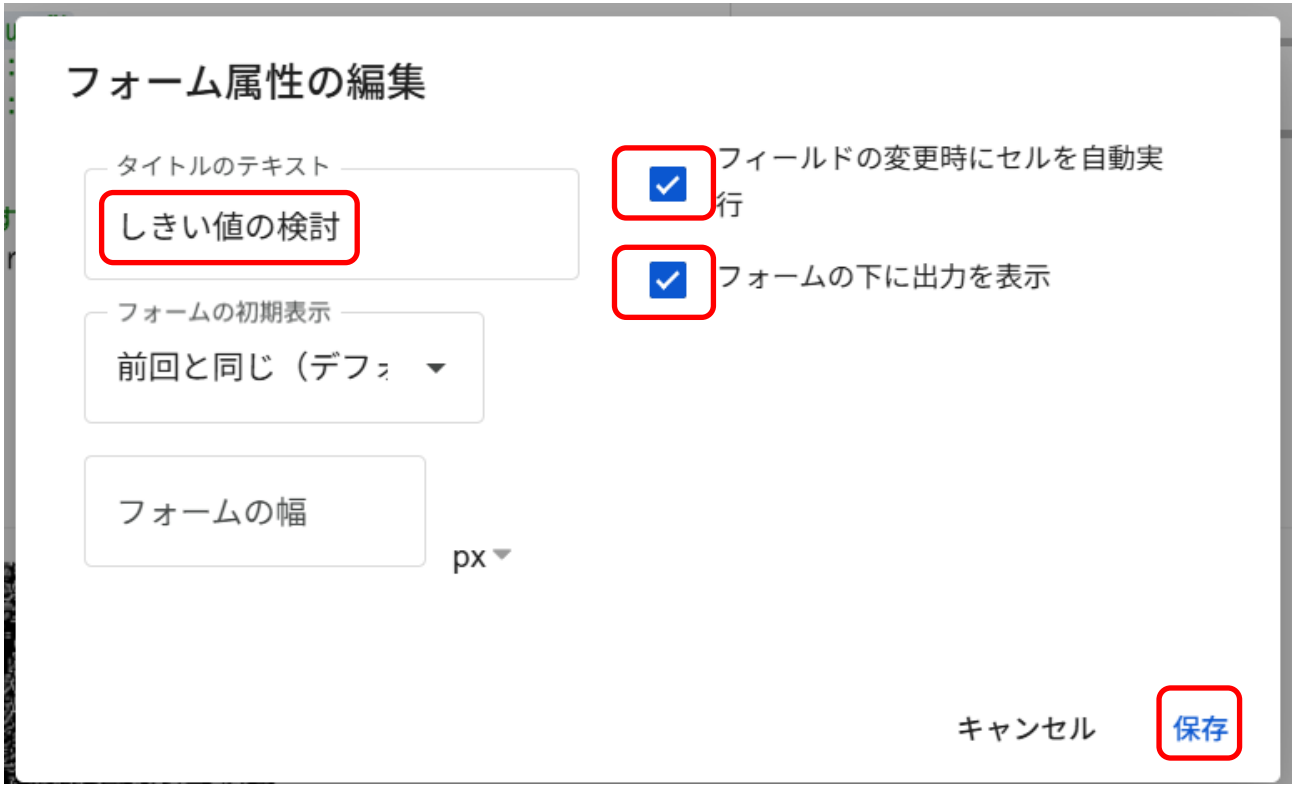
- スライダーを2つ追加（threshold1，threshold2）

（参考）しきい値を変えられるようにする

- スライダーを2つ追加（threshold1，threshold2）

（参考）しきい値を変えられるようにする

- 三点リーダー＞フォーム＞フォーム属性の編集



（参考）しきい値を変えられるようにする

- 下記のプログラムを記入して実行する

しきい値の検討

```
# @title しきい値の検討 {"run":"auto","vertical-output":true}
threshold1 = 100 # @param {"type":"slider","min":0,"max":360,"step":1}
threshold2 = 280 # @param {"type":"slider","min":0,"max":360,"step":1}

#Cannyアルゴリズムを使用してエッジ検出を行う
#theshold1,threshold2をしきい値の引数として設定する
canny_edges = cv2.Canny(gray_img, threshold1, threshold2)

# 実行結果の画像を表示する
plt.figure(figsize=(6, 6))
plt.imshow(canny_edges, cmap='gray', vmin= 0, vmax=255)
plt.axis('off')
plt.show()
```

（参考）しきい値を変えられるようにする

- スライダーでしきい値を変化させると実行結果に反映される