「画像処理基礎」

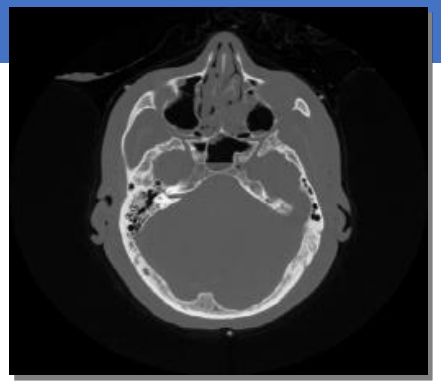
九州大学 大学院システム情報科学研究院
情報知能工学部門
データサイエンス実践特別講座
備瀬竜馬, Diego Thomas, 正井克俊

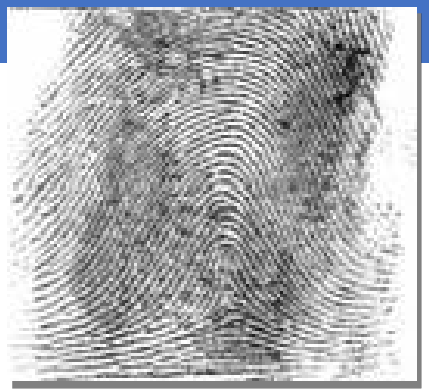
画像解析概要

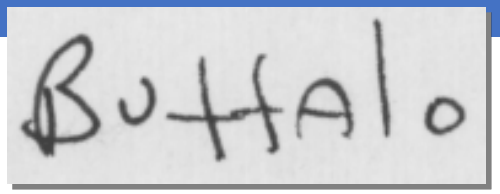
画像データ解析の流れ

画像データ
（画像空間）

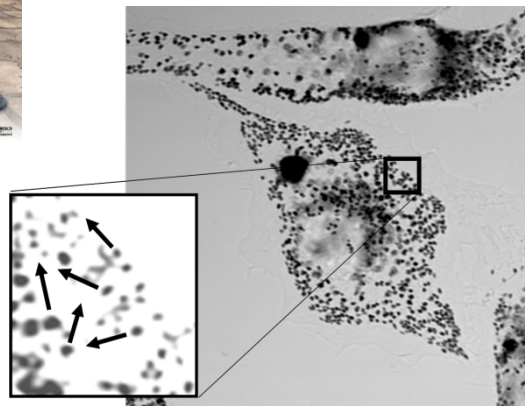
いろいろな画像






Buffalo


DreamWorks
シュレック3
FAR FAR AWAY



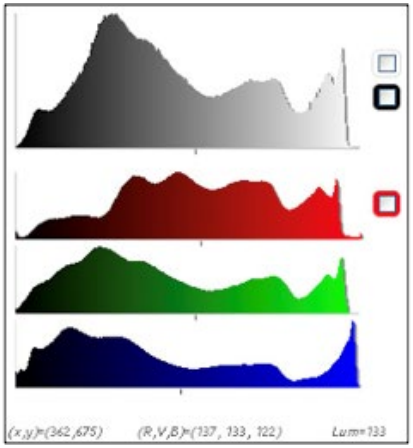

ワンセグ データ放送
いつでも！どこでも！
ニュース
スポーツ
天気予報
交通情報
ご意見・お問い合わせ

画像データは意味を持つか？

- 画像はあくまでピクセルごとの輝度値情報のみ

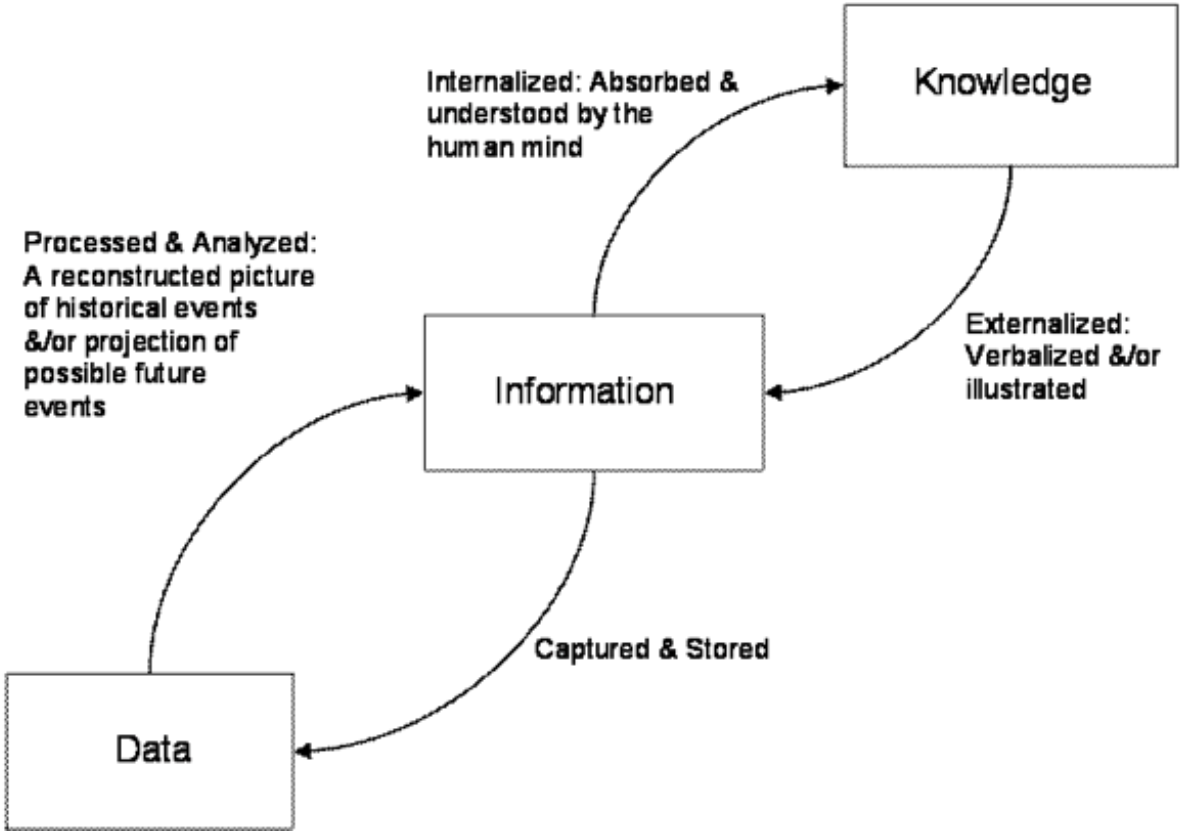
<u>意味情報</u>
着物を着た日本人女性が、三味線を弾いている
江戸時代の絵？

Computer:



The flow of analysis

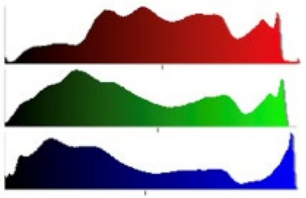
- データから情報を取り出し、情報から知識へ

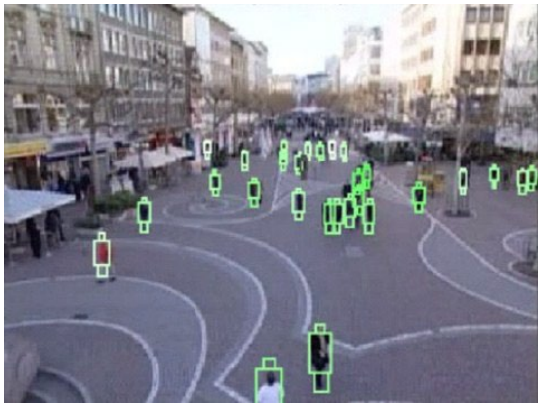


データからの情報の抽出

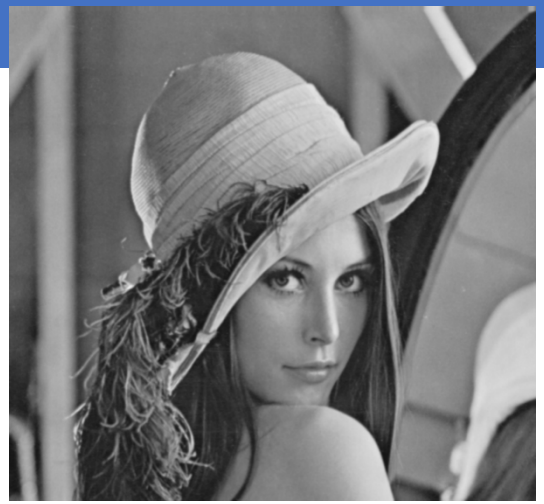
- ピクセルの数値情報でしかない画像から、人は様々な情報を抽出できる
  - どんな色が多い?
  - エッジはどこにある?
  - 人はどこにいる？
  - 建物はどこにある？

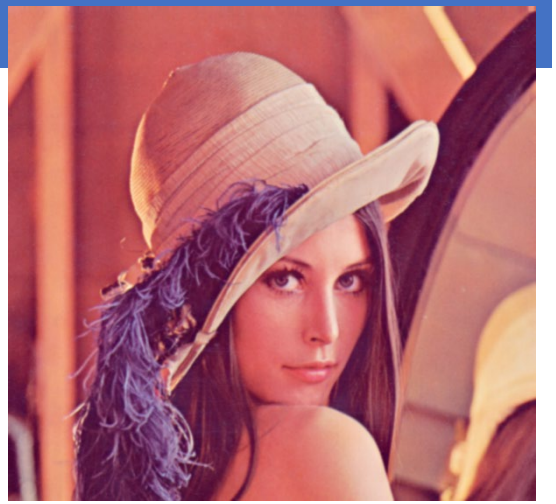
Look into histograms

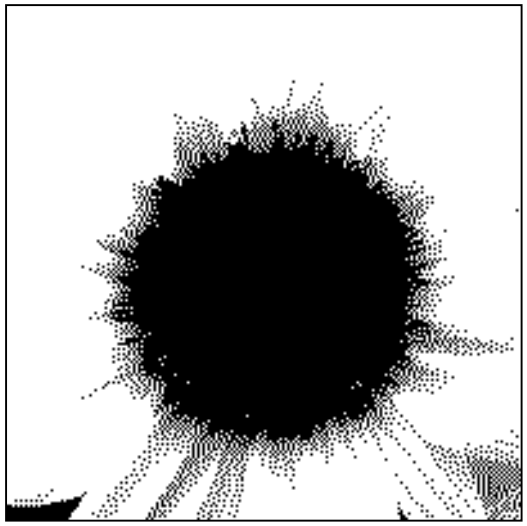




2値画像，濃淡画像，カラー画像





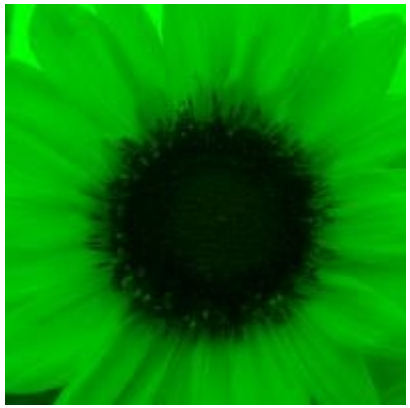


カラー画像

=

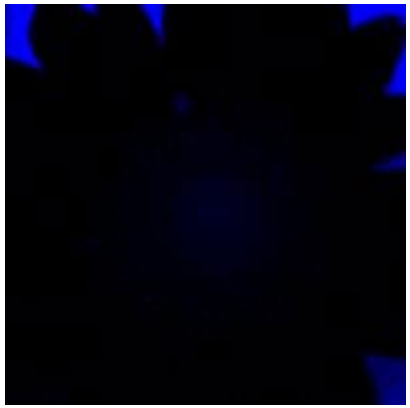
R成分

+



G成分

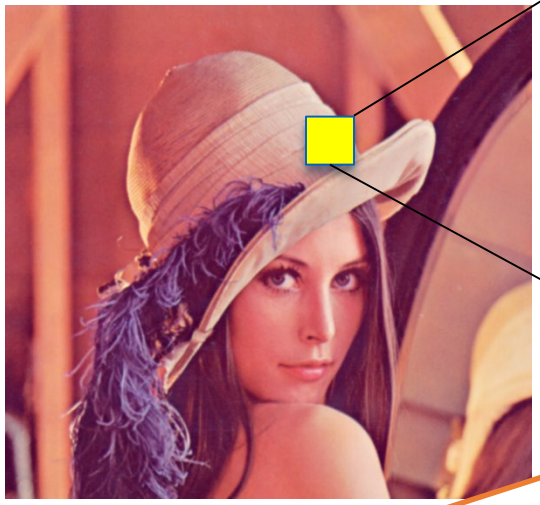
+

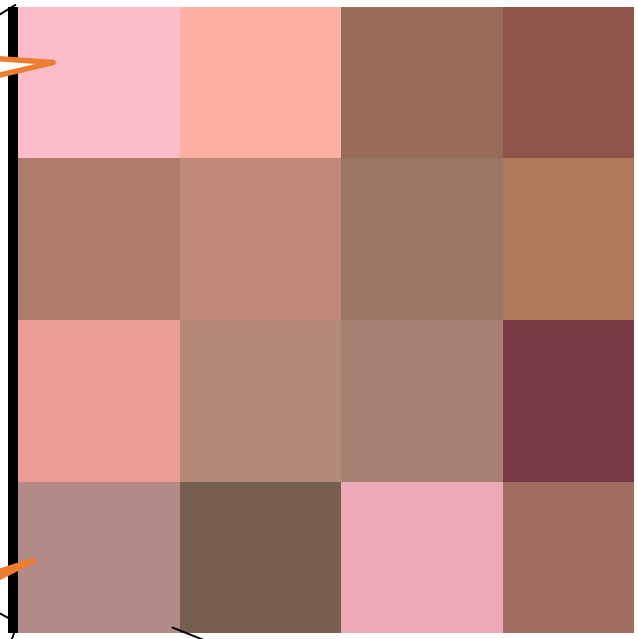


B成分

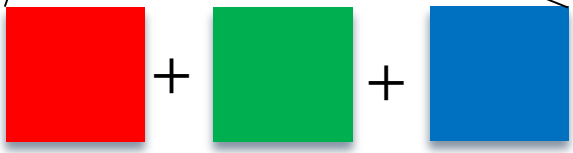
デジタル画像

規則的に並んだ
有限個の点（画素）の集合





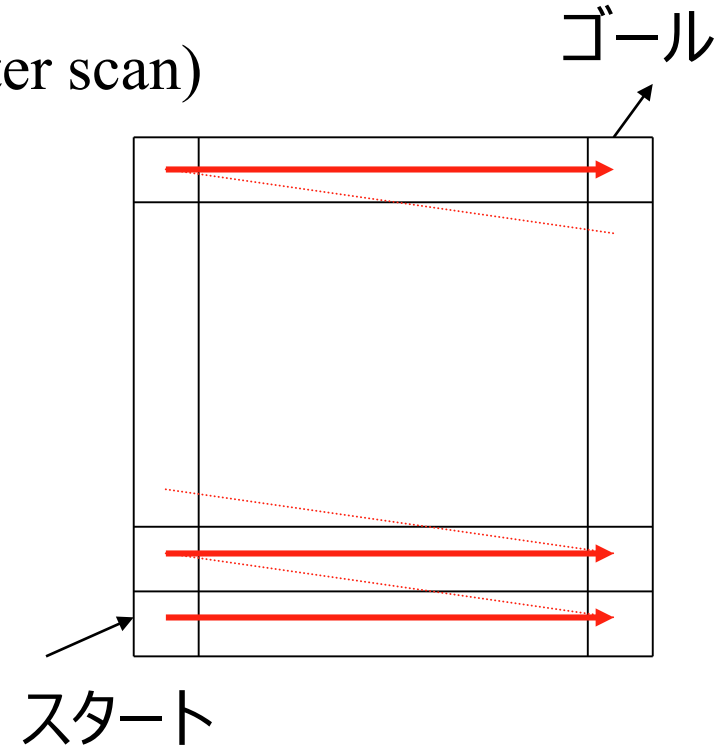
各画素の色は、
赤、緑、青の3つの成分で
表現されている



R成分 0～255　G成分 0～255　B成分 0～255

画像の走査（スキャン）

- ラスタ走査 (raster scan)



すべての画素を１回ずつ通過

ラスタ走査 (つづき)

- 画像のベクトル表現

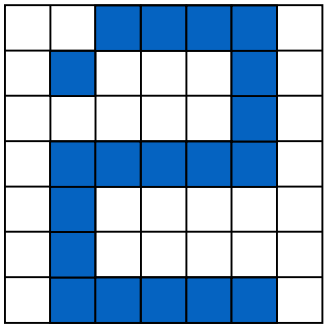
  画素値をラスタ走査順に列挙

$$\underbrace{\left(f_{1,1},\ldots,f_{1,Y},f_{2,1},\ldots,f_{x,y},\ldots,f_{X,Y}\right)}_{XY\text{次元ベクトル}}$$

- 任意の画像＝XY次元空間内の１点

画像のベクトル表現(vector representation of image)

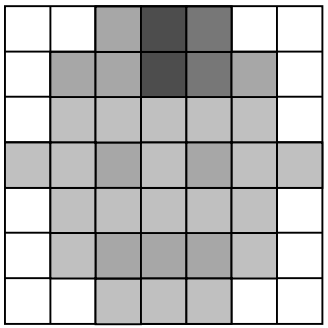
- 2値画像の場合の例



7×7画像

$$\Rightarrow x = \underbrace{(0,0,1,1,1,1,0,0,1,\ldots,1,0)^T}_{\text{49次元ベクトル}}$$

- 多値画像の場合の例

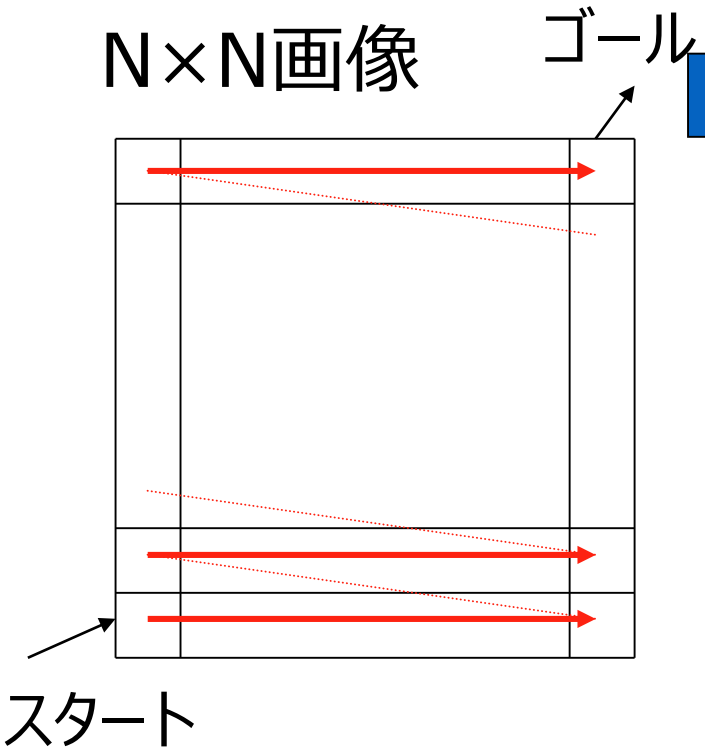


7×7画像

$$\Rightarrow x = \underbrace{(0,0,255,213,182,0,0,\ldots,0)^T}_{\text{49次元ベクトル}}$$

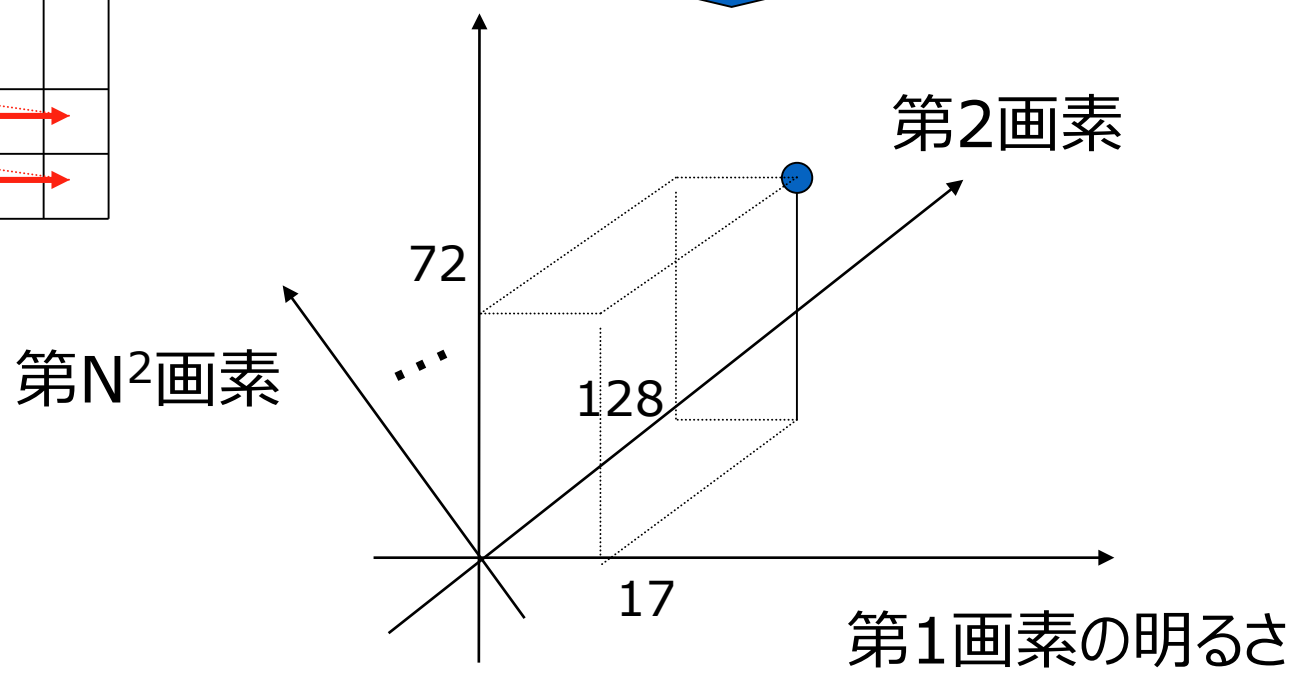
“画像空間” (image space)

N×N画像

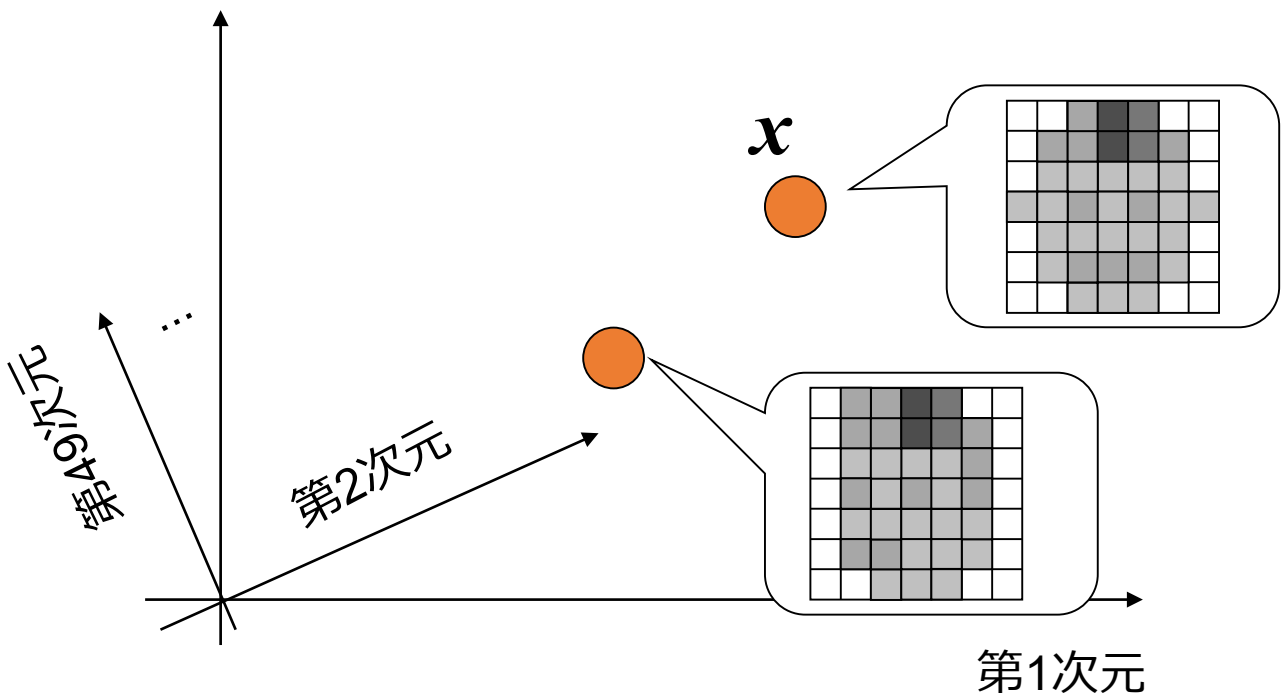


(17, 128, 72, ..., 153)

N^2次元ベクトル

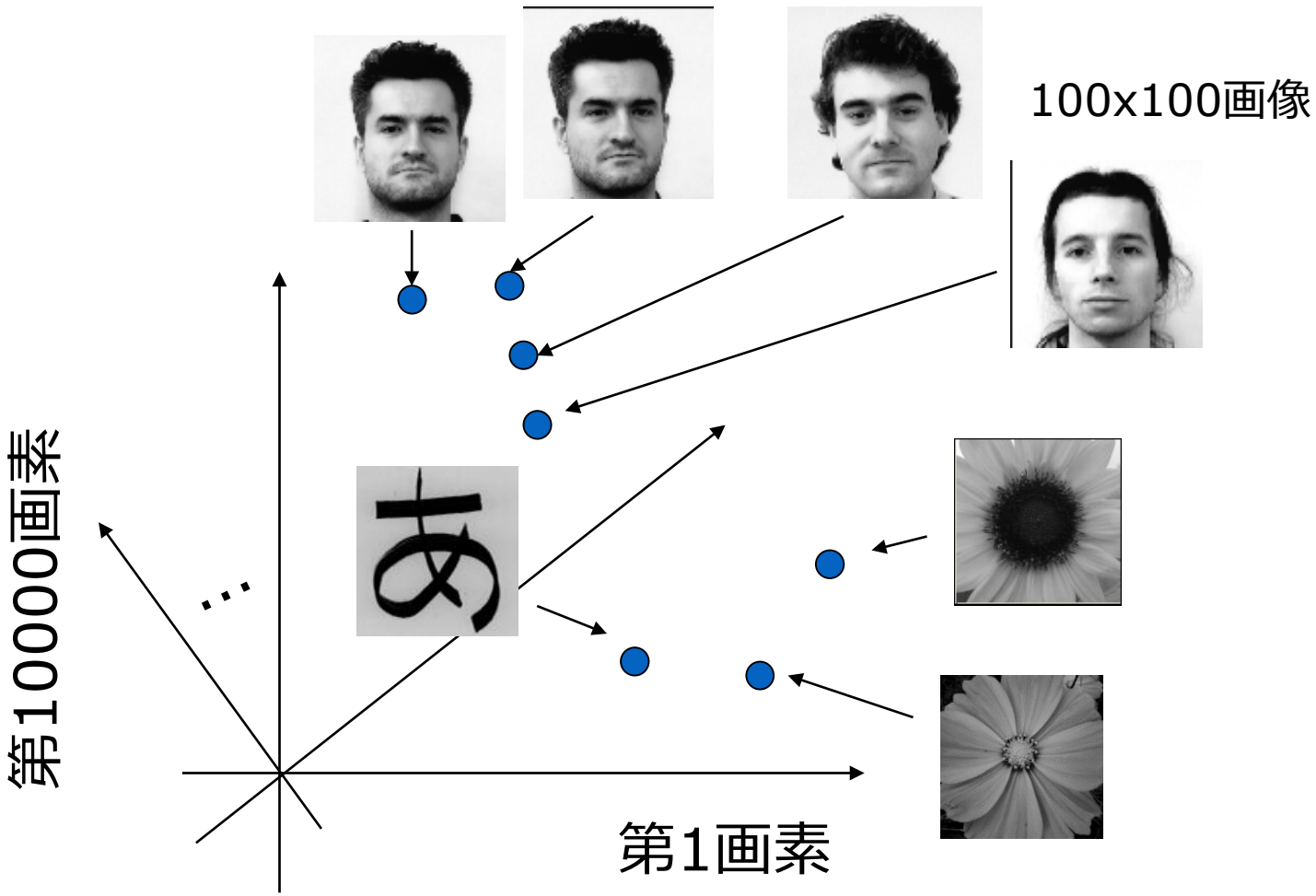


画像空間:任意の１点＝1画像



A point in the image space corresponds to an image

画像空間

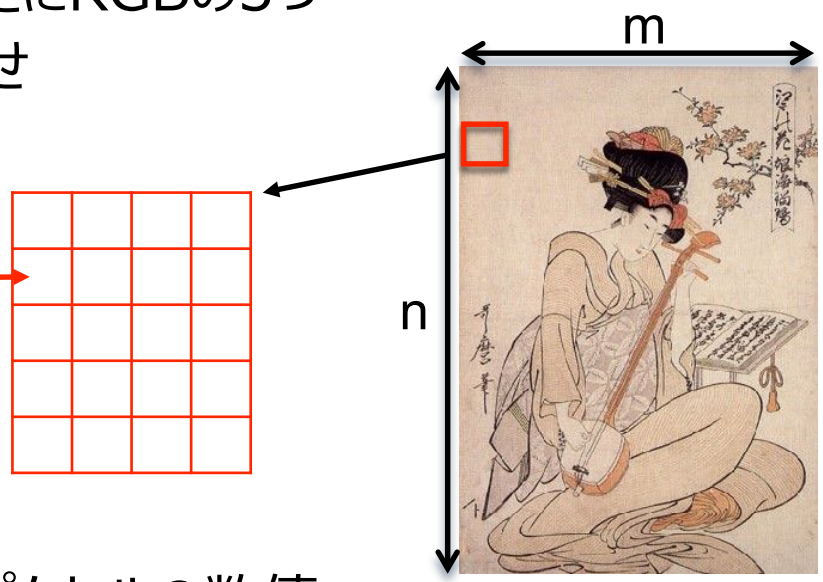


任意の100 x 100画像は10000次元空間で表せる！

画像データ

- カラー画像データは、ピクセルごとにRGBの3つのチャンネルの数字の組み合わせ

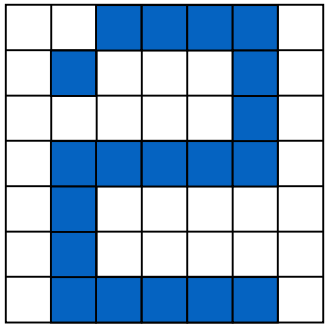
pixel = [red, green, blue]



- n 行 m 列の画像： $n \times m$ ピクセルの数値
- カラー画像：1つのピクセルは3つのチャンネルの数値を持つ
  - red, green, blue.
- 例：100×100のカラー画像
  ⇒3×100×100=30000の次元のベクトル

練習

- Numpyをインポートする



- ラスタ走査を使って “2”の画像をベクトルにする
白ピクセル＝ 0 。青ピクセル＝255

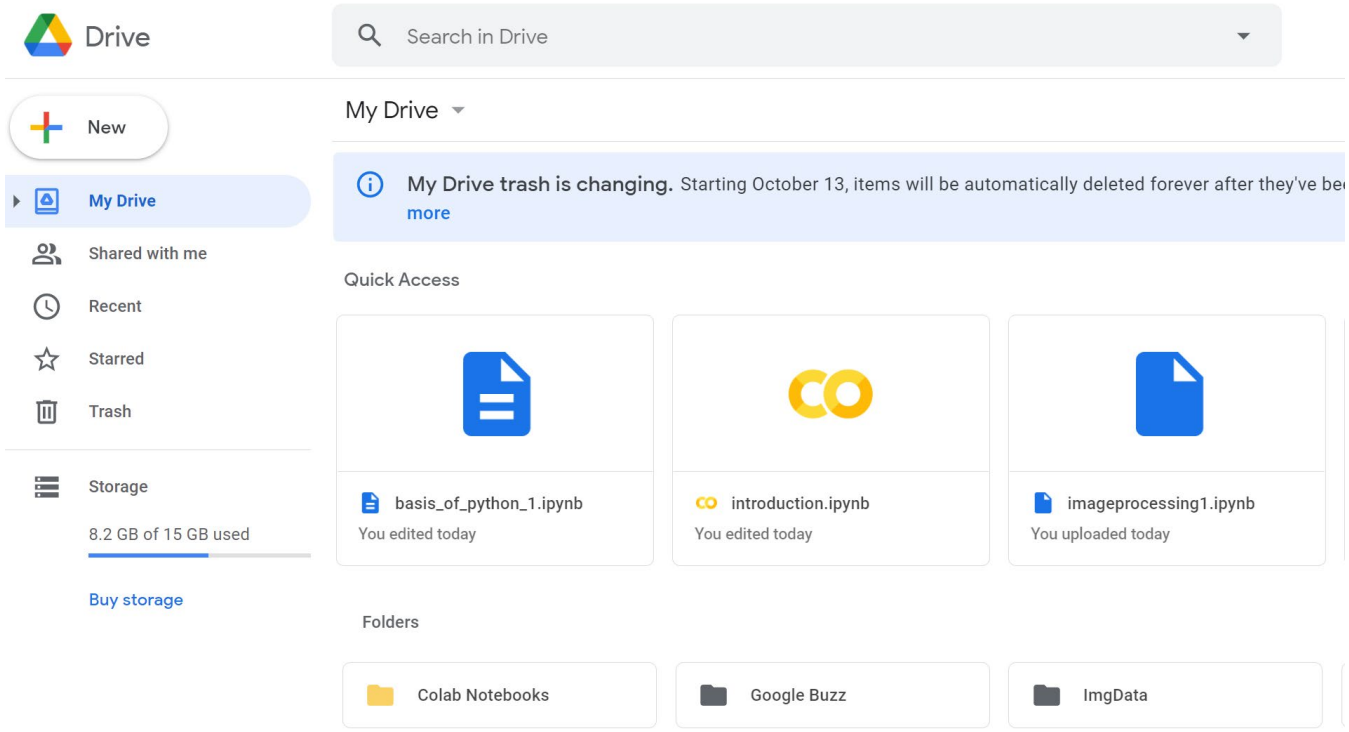
- numpy.array を使ってデータ変換する

- Reshape を使ってベクトルから７ｘ７の行列を求める

- ７ｘ７の行列をプリントする

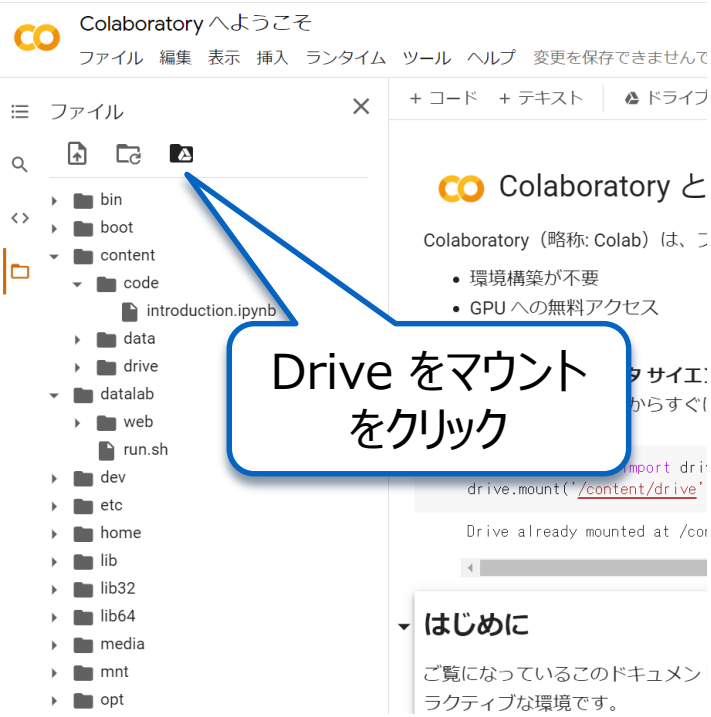
Google Driveとの連携

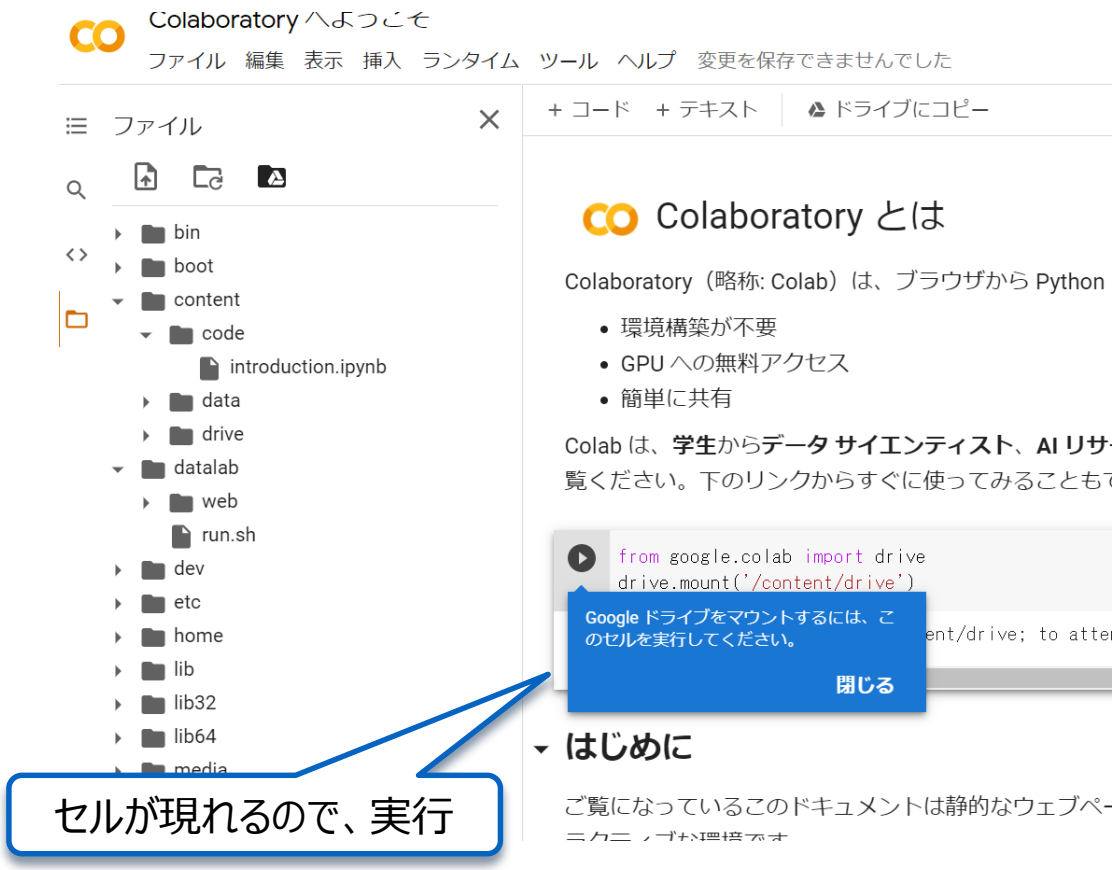
準備：Google Driveへフォルダごとアップ



フォルダごとドラッグすれば、
フォルダの中身も全てアップロード可能

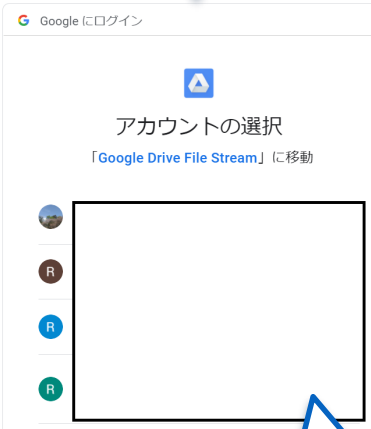
Google Driveとの接続





Google Driveとの接続

リンクをクリック

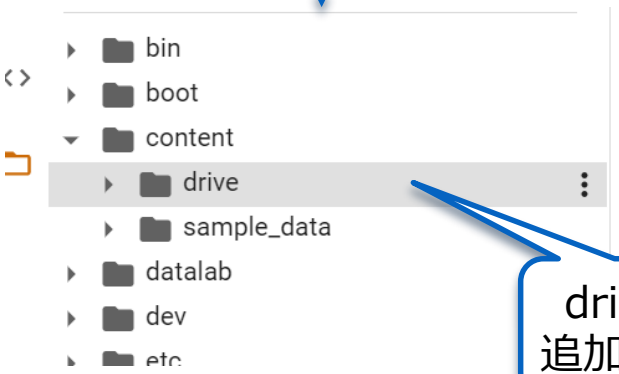


Google Driveの
アカウントを選択

許可

コピー

ペースト

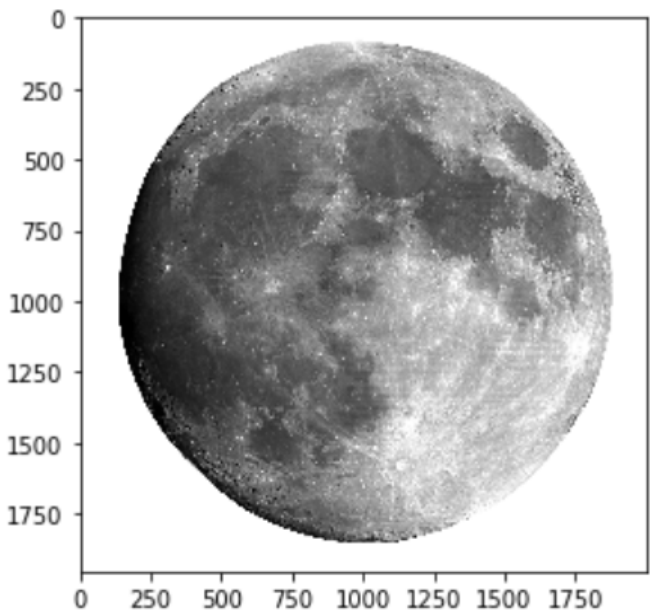


driveが
追加された

Drive上のファイルを読めるかを確認

```
from skimage import io
from skimage import data
#data_dir = './ImgData/'
data_dir = './drive/My Drive/ImgData/'
filename = data_dir + 'moon.png'
moon = io.imread(filename)
io.imshow(moon)
io.show()
```

Google Drive上のファイルのパスを記載



画像解析に便利なライブラリ

OpenCV / scikit-image

OpenCV

If interested, try at home ☺

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

- http://opencv.org
- CV = “Computer Vision”
- 様々な画像解析ライブラリ（関数群）が提供されている
- Python用にライブラリが整備されている
- 注）アナコンダにはデフォルトでは入っていないため、インストールが別途必要。

scikit-image

- Python用画像処理ライブラリ
- Numpy arraysを利用
  - グレースケール画像は2次元配列(array)で表現
- http://scikit-image.org

参考: http://www.scipy-lectures.org/packages/scikit-image/

データの取得

- skimage.dataに予め用意されている画像を読み込み

```
from skimage import data
coins = data.coins()
```

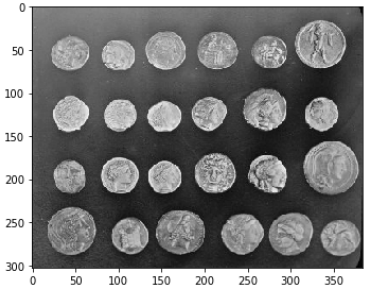
- ファイルから画像を読み込み：skimage.io.imread()

```
from skimage import io
filename = 'moon.png'
moon = io.imread(filename)
```

画像の可視化

- 画像の表示：skimage.io.imshow()

```
io.imshow(coins)
```



- 画像サイズ

```
print("Dimensions of the coin image: ", coins.shape  )
print("Dimensions of the moon image: ", moon.shape   )
```

```
Dimensions of the coin image:  (303, 384)
Dimensions of the moon image:  (1955, 2000, 4)
```

- 輝度値の最大、最小、平均

```
print("Minimum value for image coins: ", coins.min() )
print("Maximum value for image coins: ", coins.max() )
print("Average value for image coins: ", coins.mean() )
```

```
Minimum value for image coins:  1
Maximum value for image coins:  252
Average value for image coins:  96.8555160204
```

練習1

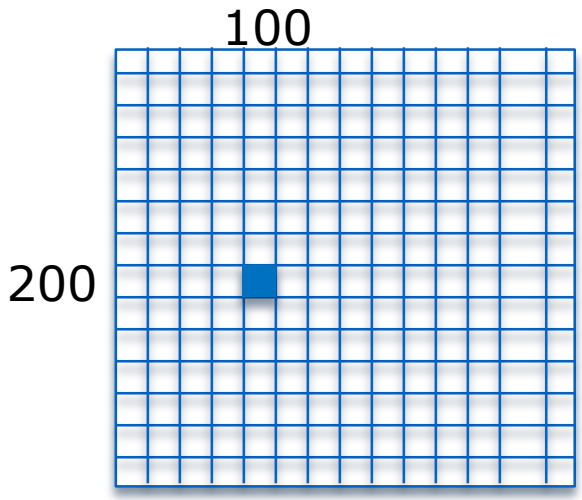
- skimage から io をインポートする
- “coffee.tiff” を読み込んで、可視化する
- 画像のサイズとピクセルの寸法を求める

画像と配列，画素レベルの処理

画像データ構造，動画，閾値処理

画素値へのアクセス

- 画素値へのアクセス




```
print("pixel value of image coins at location (200,100): ", coins[200,100] )
print("pixel value of image moon at location (100,300): ", moon[100,300]   )

coins[200,100] = 0
print("New value of image coins at piel (200,100): ", coins[200,100] )
```

```
pixel value of image coins at location (200,100):  165
pixel value of image moon at location (100,300):  [255 255 255   0]
New value of image coins at piel (200,100):  0
```

“moon” は 4 つの値を持つ! ：red, green, blue, alpha （透過度）

画素値へのアクセス

- (0,0) ：画像の左上

※）pythonでは、0 から始まる

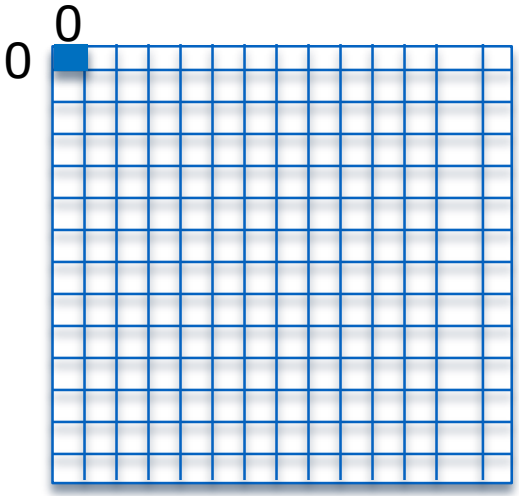


Image type	coordinates
2D grayscale	(row, col)
2D multichannel (RGB)	(row, col, ch)
3D grayscale	(pln, row, col)
3D multichannel	(pln, row, col, ch)

row ： 行
col ： 列

カラー画像

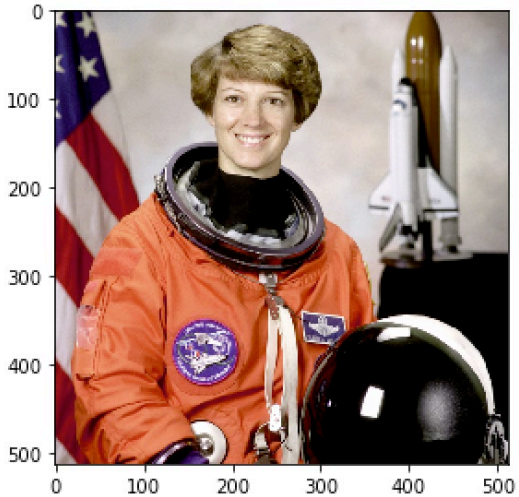
- 1画素につき、(Red,Green,Blue）の3チャンネルの数値
- データ名.shape を使って確認しよう
- それぞれのチャンネルに、0 から 255 までの値が入っている
  - 例：赤（255, 0, 0）
- Pythonで, それぞれのチャンネルにアクセスするには
  [row, col, 0 or 1 or 2]

Red　Green　Blue

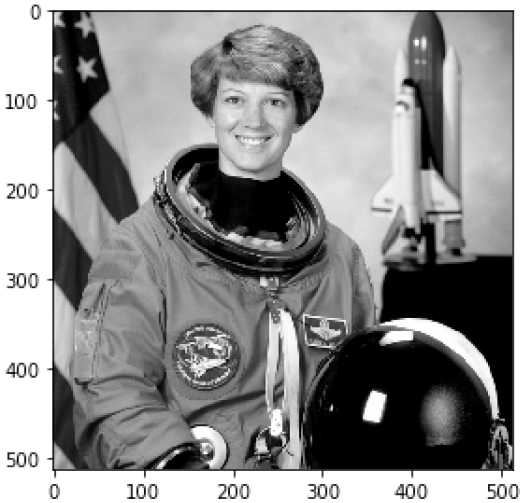
カラー画像⇔グレースケール画像変換

- （red, green, blue）⇒ gray value
- 画像変換：rgb2gray

```
from skimage.color import rgb2gray # Load the function for conversion
from skimage import data
from skimage import io
img = data.astronaut() # load the input image
img_gray = rgb2gray(img) # convert to gray image
io.imshow(img)
io.imshow(img_gray)
```



Input color image

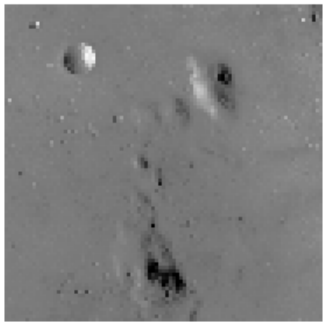


Output gray image

コントラスト強調

- 画素値： (一般的には)0 ～ 255 を取りうる
- 実際には、その中の狭い範囲にしか値がない画像も多い
  - 下記の画像では [50, 100]の値しかない
  - コントラストが低い

Low contrast image



- skimage.exposure ：画像のコントラストを強調（利用する画素値の幅を広める）する関数が提供されている

コントラスト強調手法

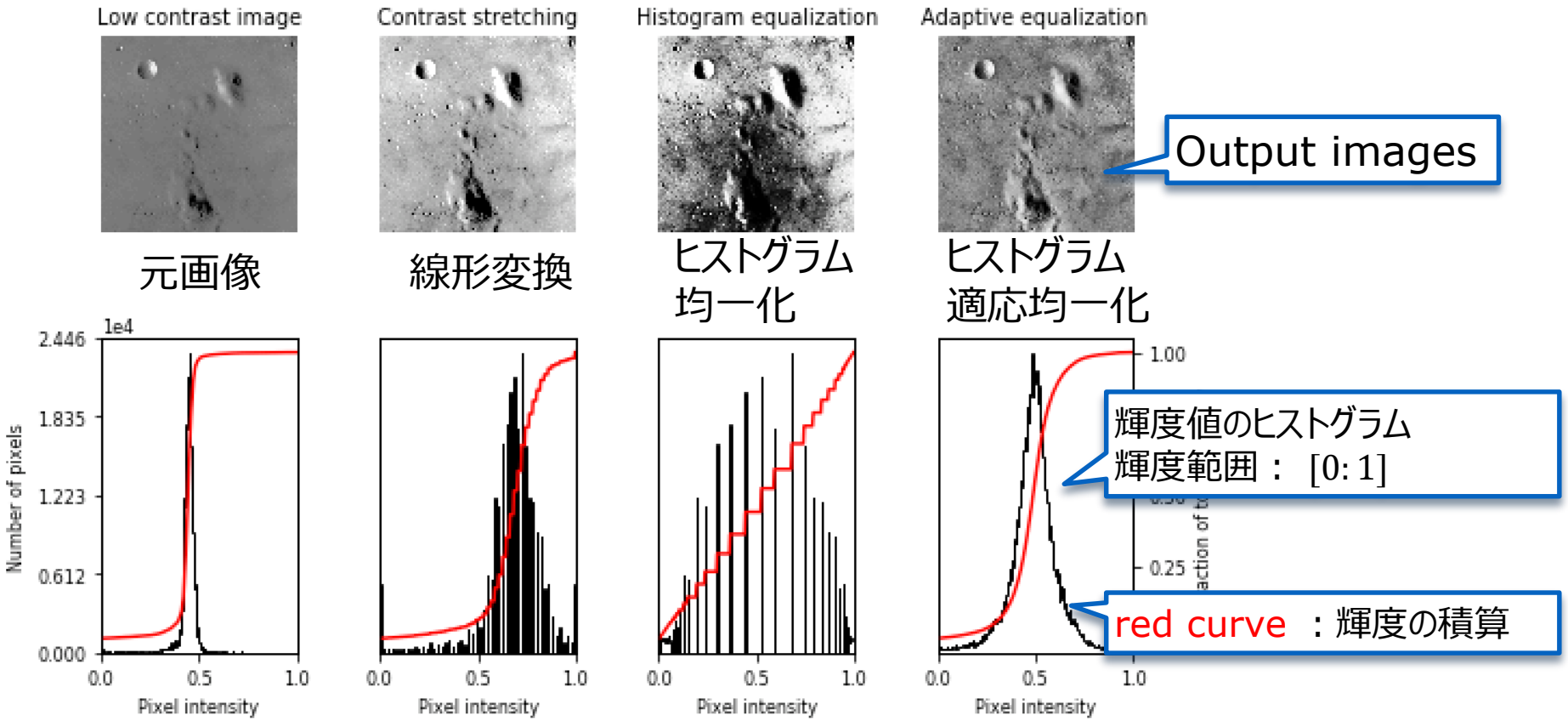
- ガンマ補正
  - 暗い画素を持ち上げる（明るくする）

- 輝度ヒストグラムを用いたコントラスト補正
  - 輝度の分布を用いて、輝度値を補正し、コントラストを強調

- 線形濃度変換
  - 輝度値の分布範囲を[0, 255]に広げる

Histogram equalization

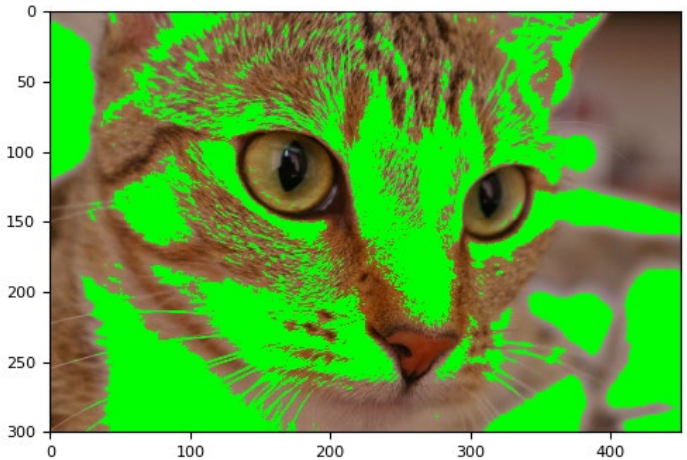
- “Histogram_Equalization.py” を利用



Source: http://scikit-image.org/docs/stable/auto_examples/color_exposure/plot_equalize.html#sphx-glr-auto-examples-color-exposure-plot-equalize-py

練習2

- skimage から data をインポートする
- “chelsea"を読み込んで、red channel > 160 となっている画素を green [0, 255, 0]に上書きして、表示




```
cat = data.chelsea() # load the cat image
reddish = cat[:, :, 0] > 160
```

- Note on data type

Data type	Range
uint8	0 to 255
uint16	0 to 65535
uint32	0 to 232
float	-1 to 1 or 0 to 1
int8	-128 to 127
int16	-32768 to 32767
int32	-231 to 231 - 1

画素レベルの処理

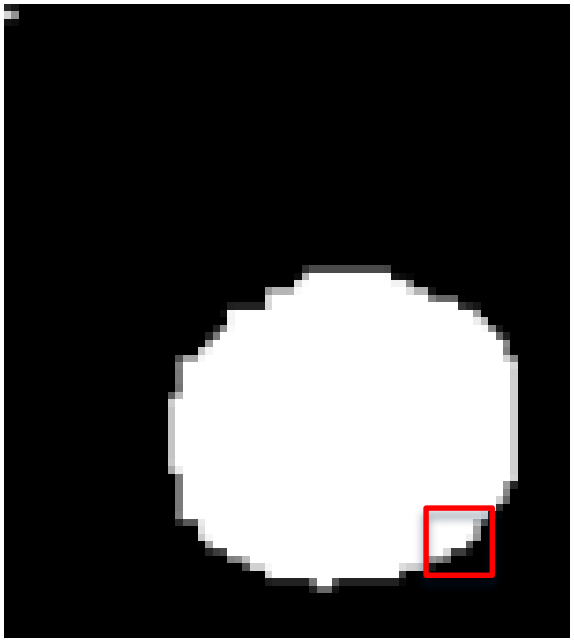
Segmentation

セグメンテーション

- ピクセルごとに背景か前景かをラベル付け

元画像

セグメンテーション画像



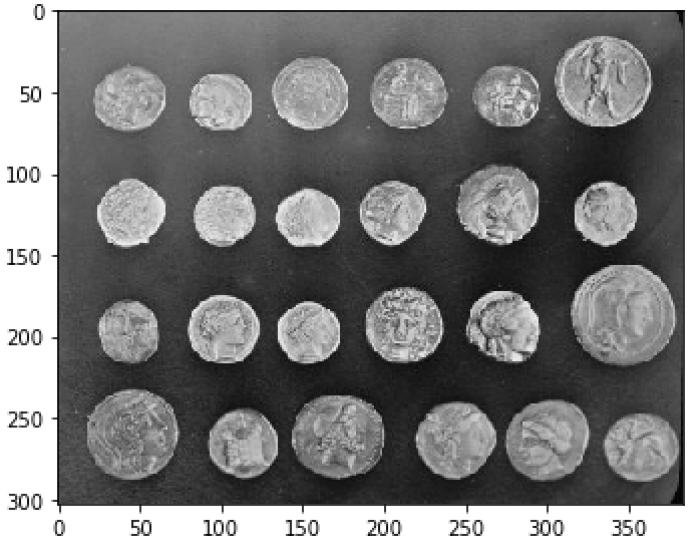
ピクセル
前景： 1
背景： 0

1	1	1	1	1	1	0
1	1	1	1	1	0	0
1	1	1	1	1	0	0
1	1	1	1	0	0	0
1	1	1	0	0	0	0
1	1	0	0	0	0	0
0	0	0	0	0	0	0

Load the coin image

- skimage.data

```
import numpy as np
from skimage import data, io
coins = data.coins()
io.imshow(coins)
```



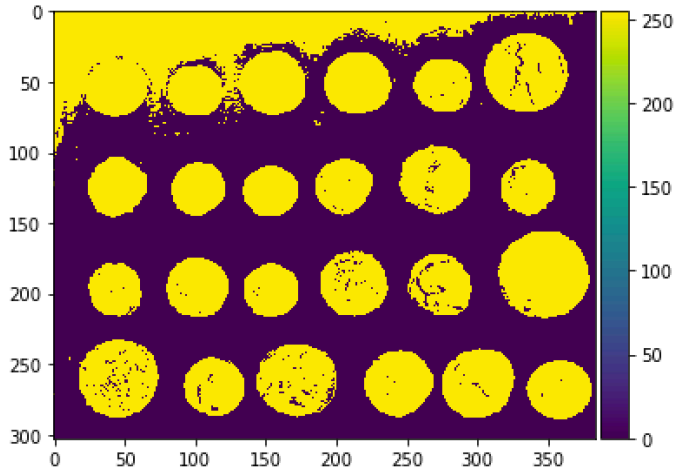
閾値によるセグメンテーション

- 閾値法：画素が閾値より高ければ前景、低ければ背景

- Example:

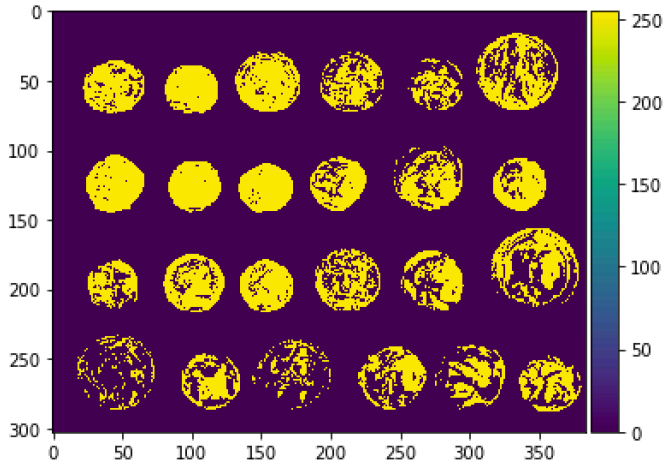
閾値：100

```
segmented_img = np.zeros(coins.shape)
mask = coins[:,:] > 100
segmented_img[mask] = 255
io.imshow(segmented_img)
```

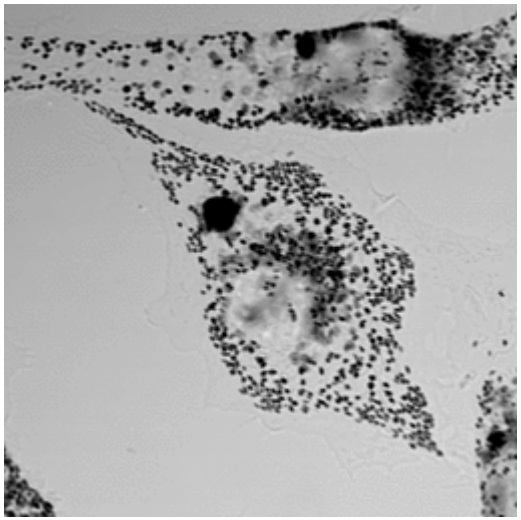


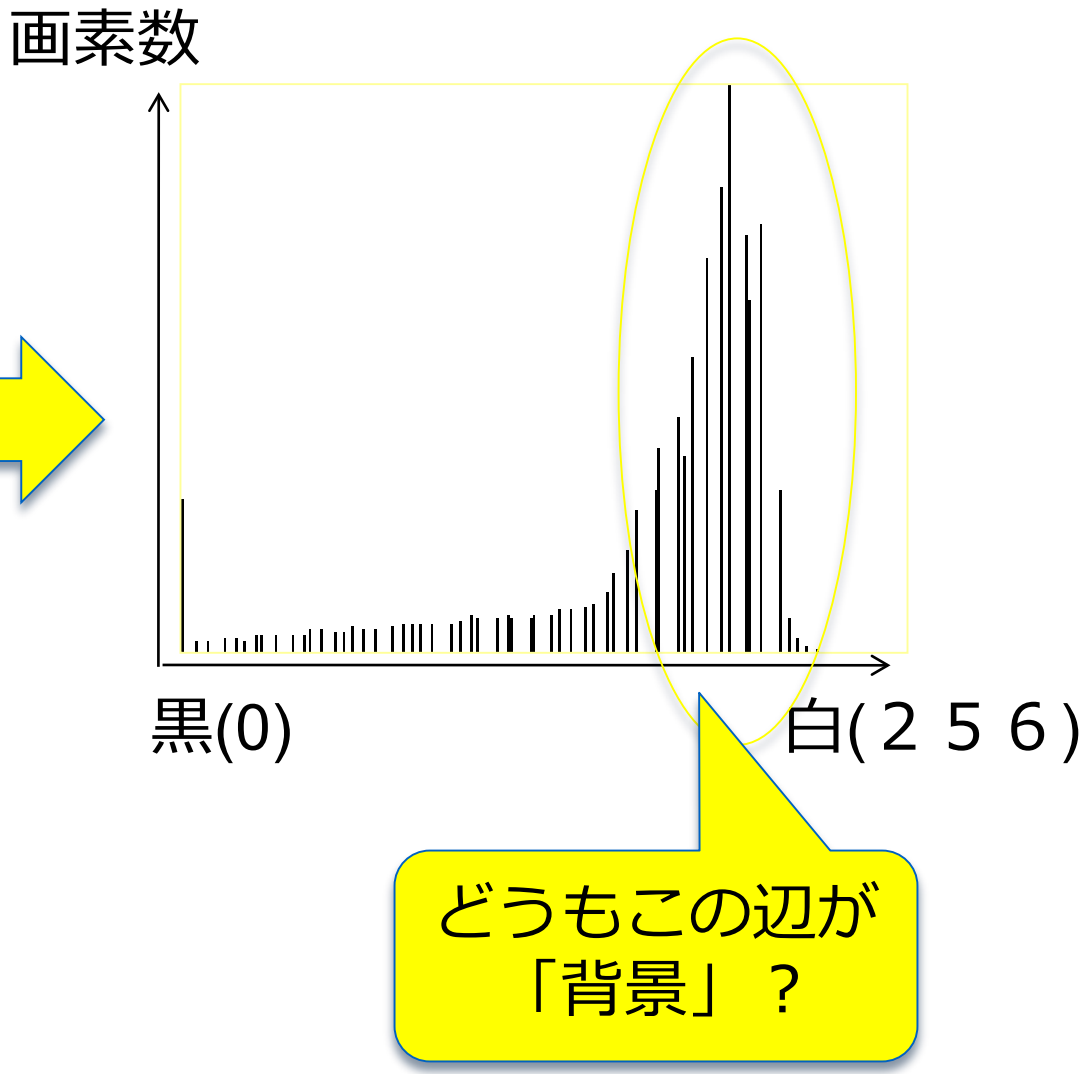
閾値：150

```
segmented_img = np.zeros(coins.shape)
mask = coins[:,:] > 150
segmented_img[mask] = 255
io.imshow(segmented_img)
```

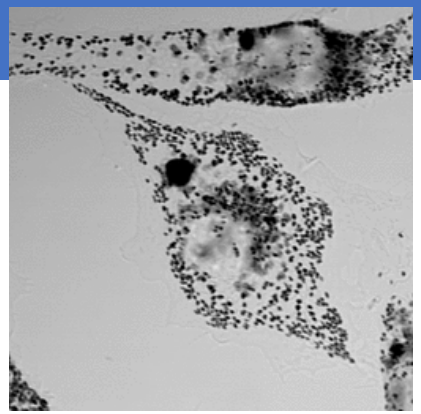


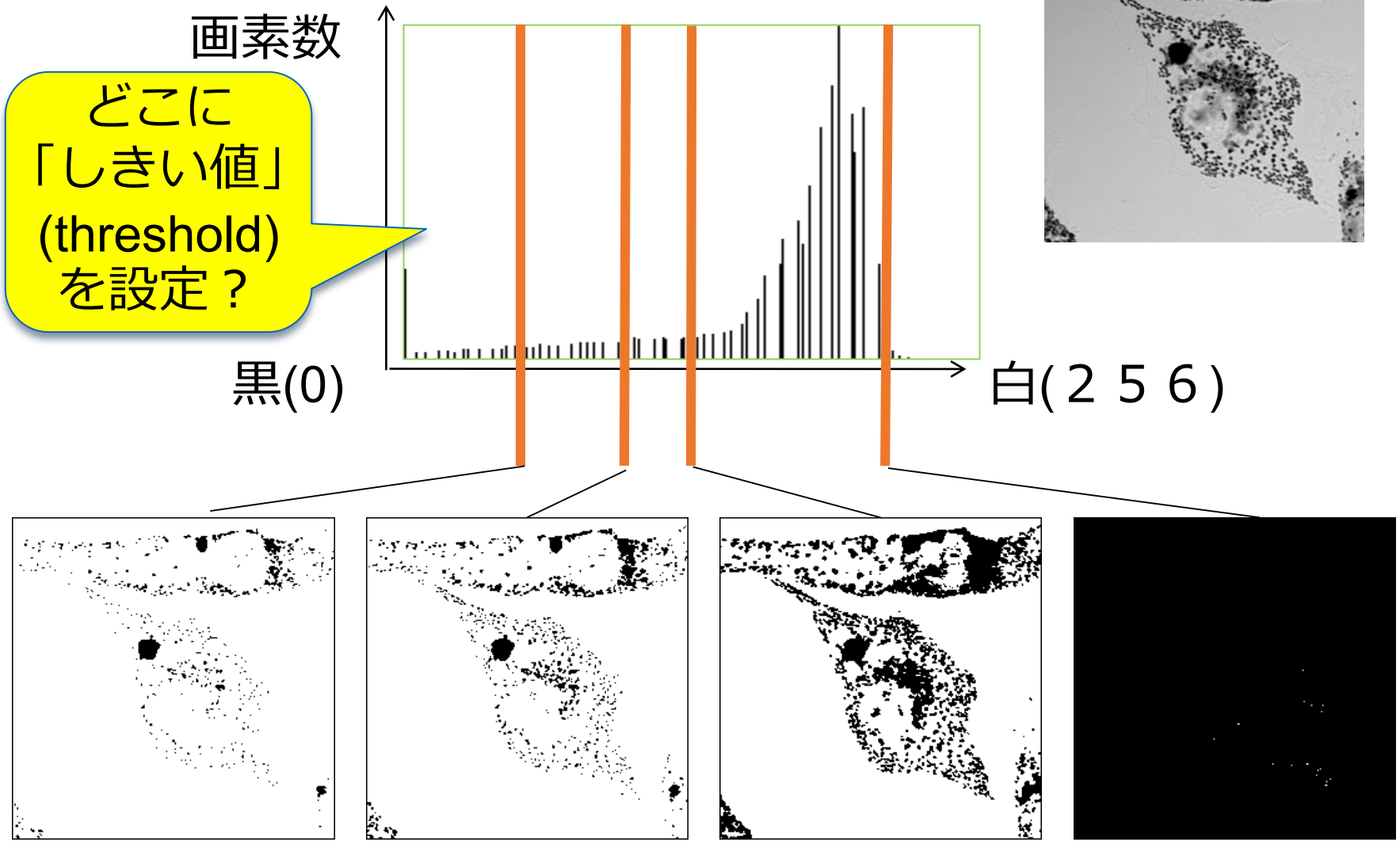
2値化の手がかり：ヒストグラム





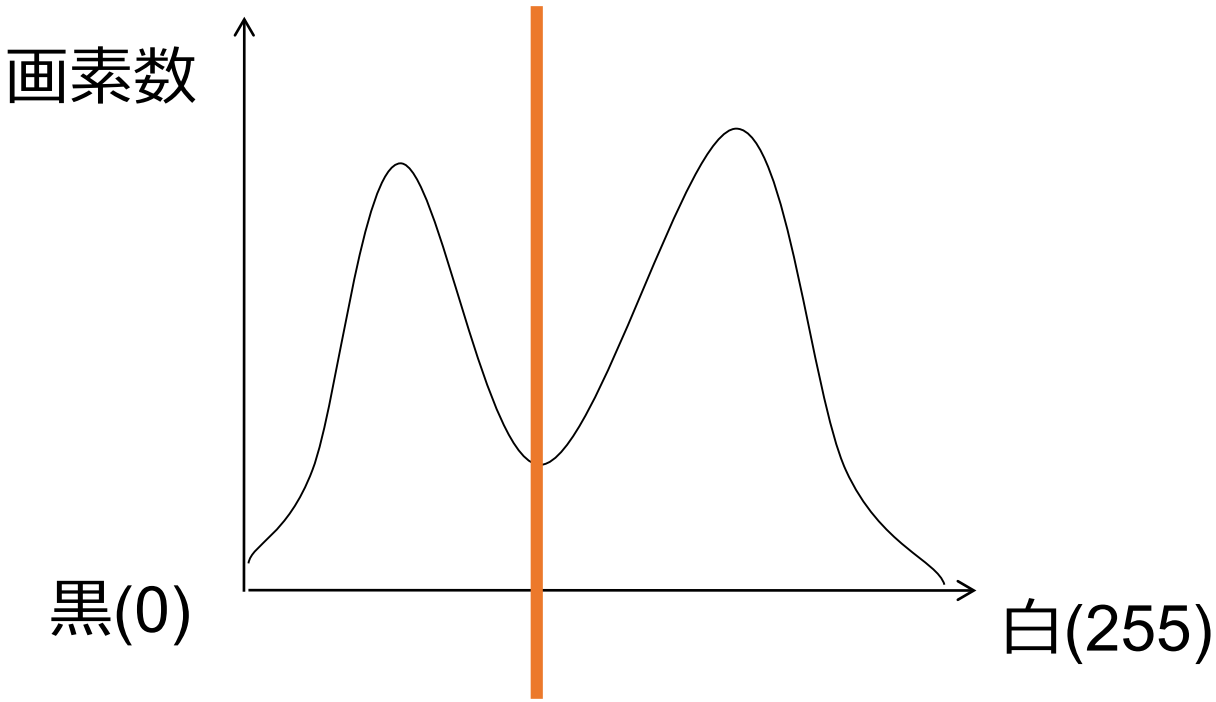
2値化の手がかり：ヒストグラム





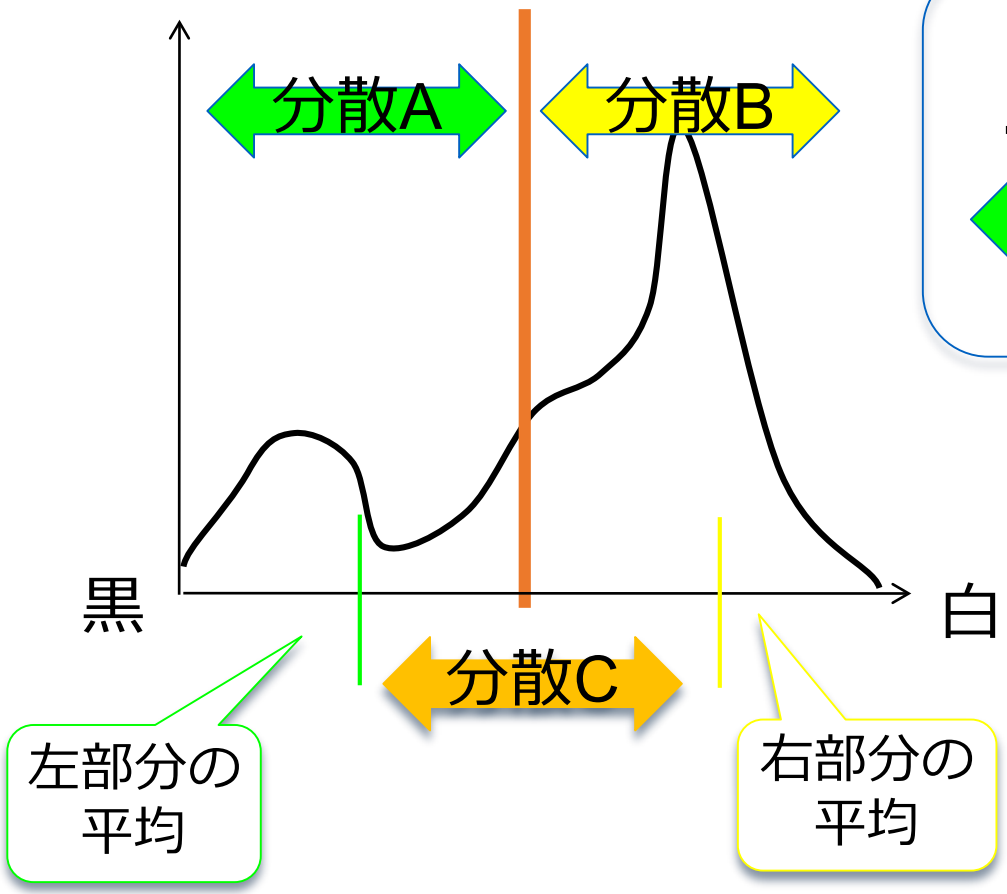
モード法

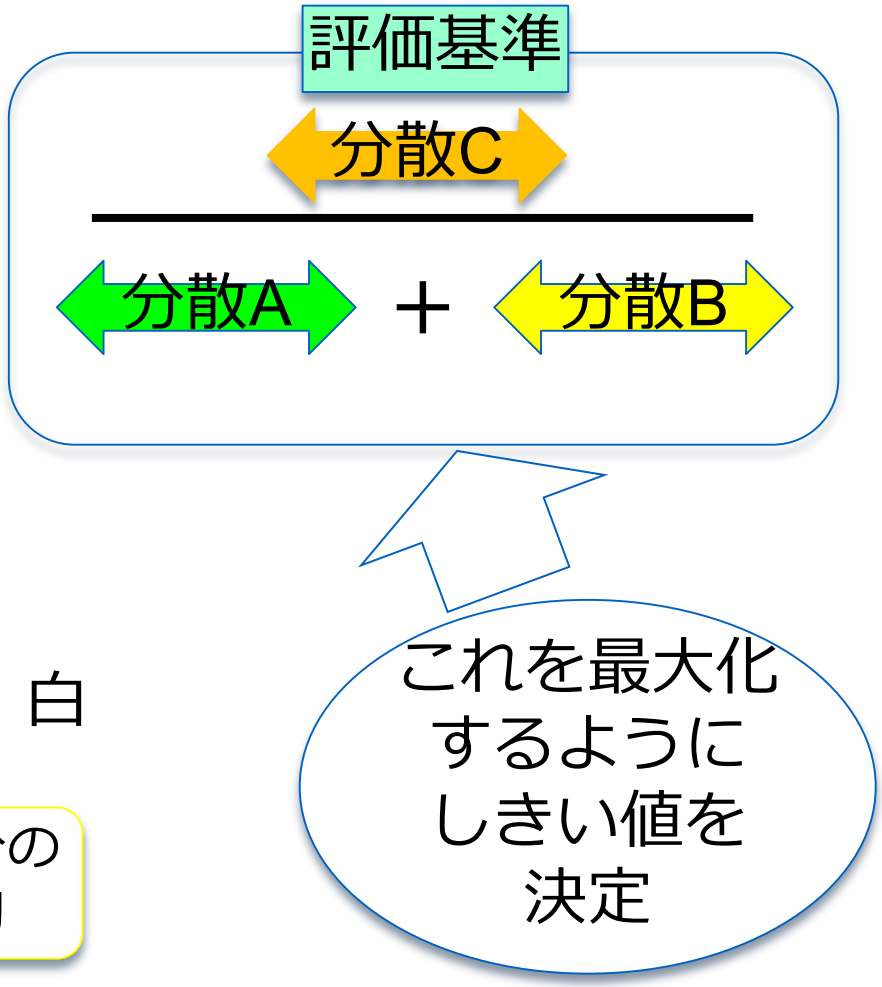
- ヒストグラムの谷＝しきい値



大津の二値化

- 最適しきい値を決定

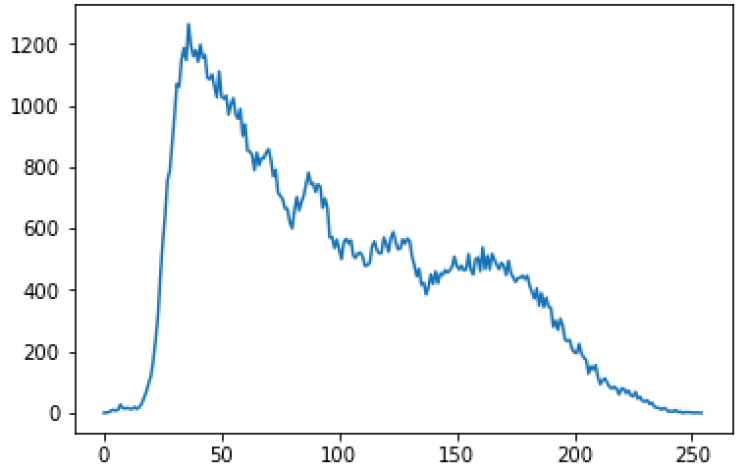


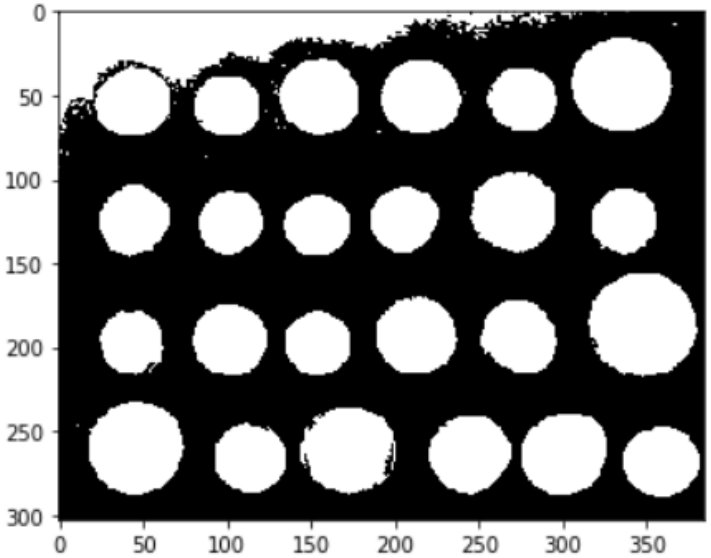


閾値の限界

- 輝度値の分布が、前景と背景で必ずしもきれいに分かれているとは限らない
  - 背景でも、前景より輝度値が高い画素（もしくはその逆）が存在
- 高すぎる閾値は、前景を取り逃し、低すぎる閾値は多めに前景を抽出してしまう

```
import matplotlib.pyplot as plt
histo = np.histogram(coins, bins=np.arange(0, 256))
plt.plot(histo[1][0:255], histo[0])
```

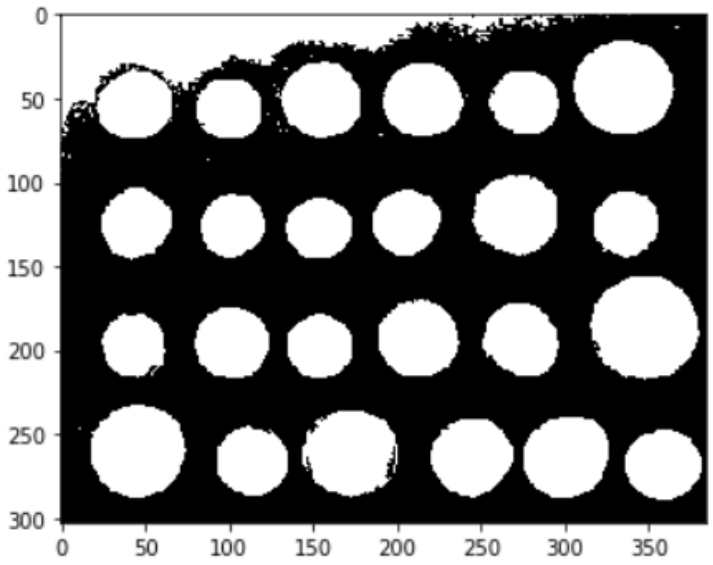




Filling the holes

- 境界の内側をぬりつぶすことで、コインの領域を抽出
- ndi.binary_fill_holes()を利用

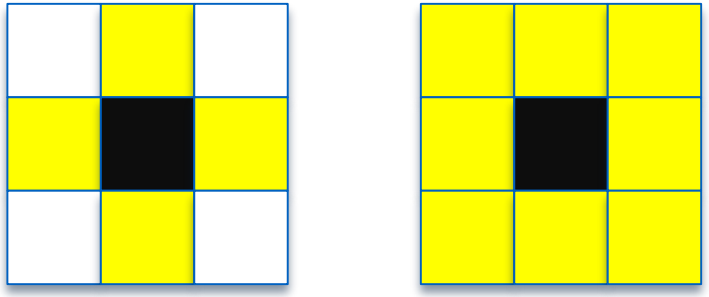
```
from scipy import ndimage as ndi
fill_coins = ndi.binary_fill_holes(edges)
io.imshow(fill_coins)
```



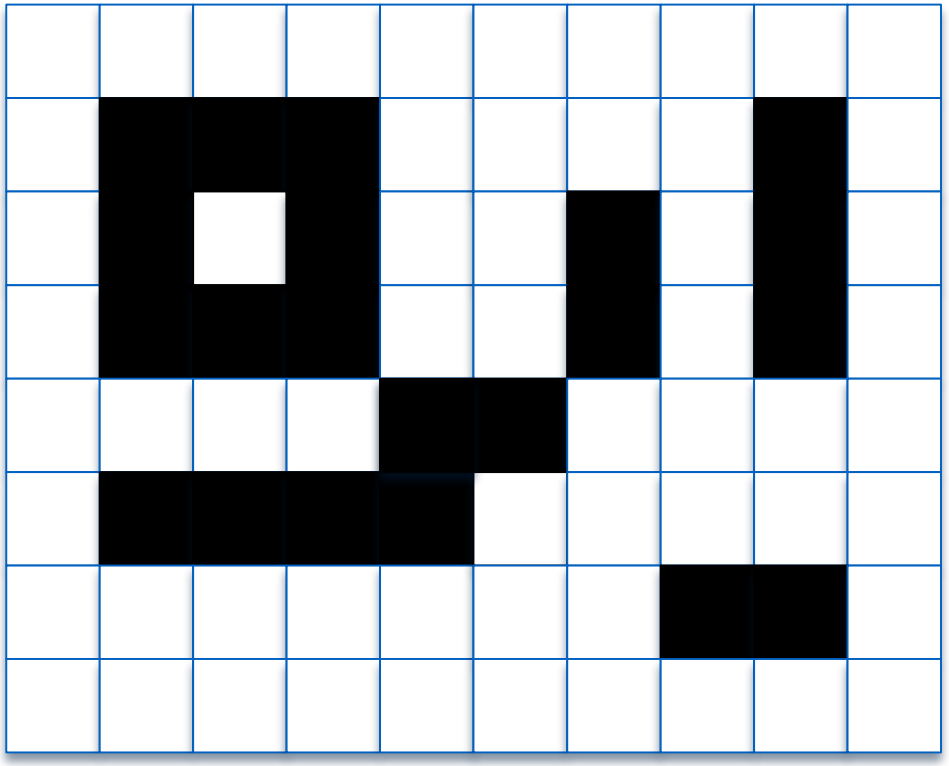
連結成分解析

- 連結成分＝黒画素（もしくは白画素）の「塊」

- ４連結と8連結



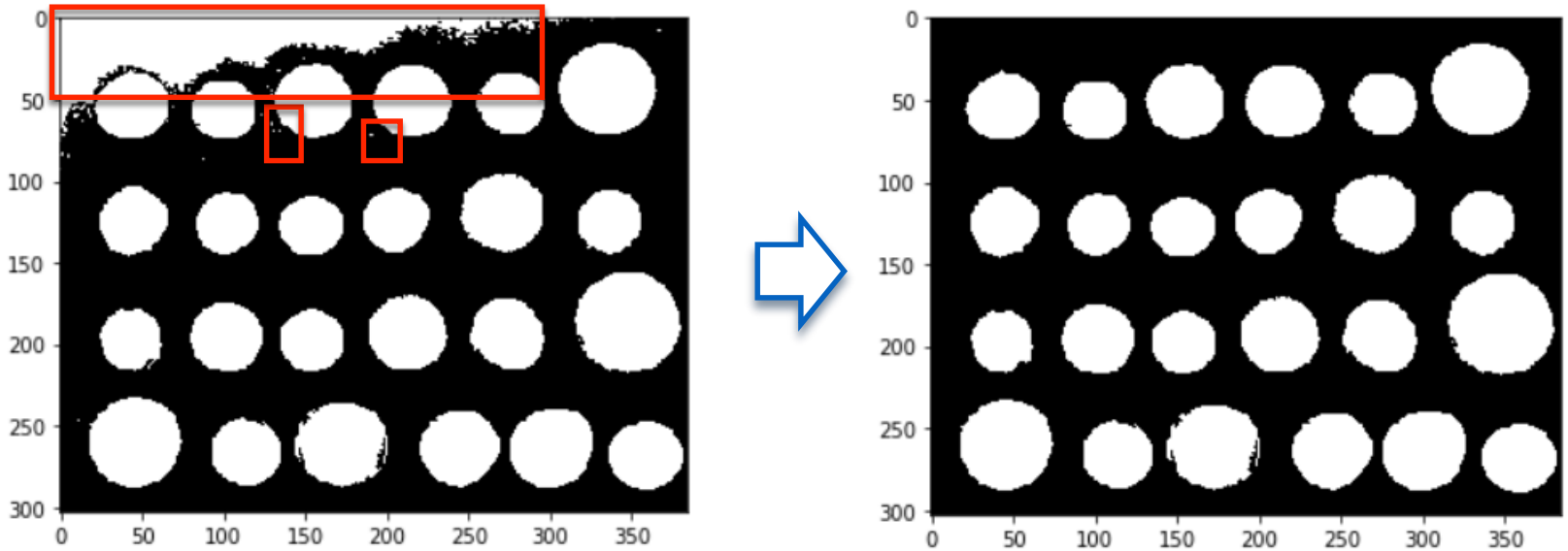
４連結だと何個の連結成分？8連結では？



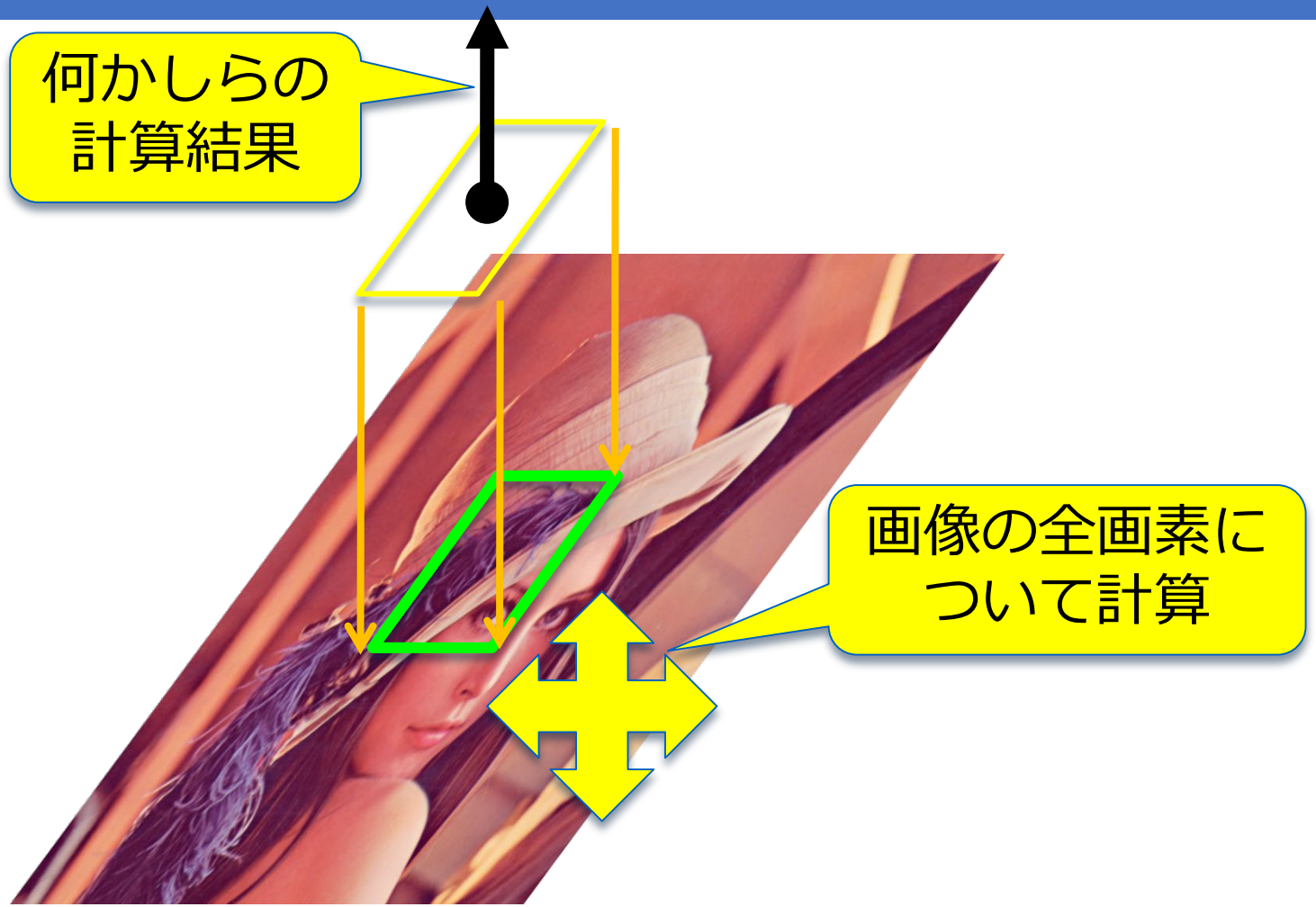
Removing small objects

- ラベリングされた小さすぎる、大きすぎるオブジェクトは間違って抽出された可能性が高いので、大きさをチェックして、削除
- ndi.label function を利用

```
label_objects, nb_labels = ndi.label(fill_coins)
sizes = np.bincount(label_objects.ravel())
mask_sizes = sizes > 20
mask_sizes[0] = 0
coins_cleaned = mask_sizes[label_objects]
io.imshow(coins_cleaned)
```

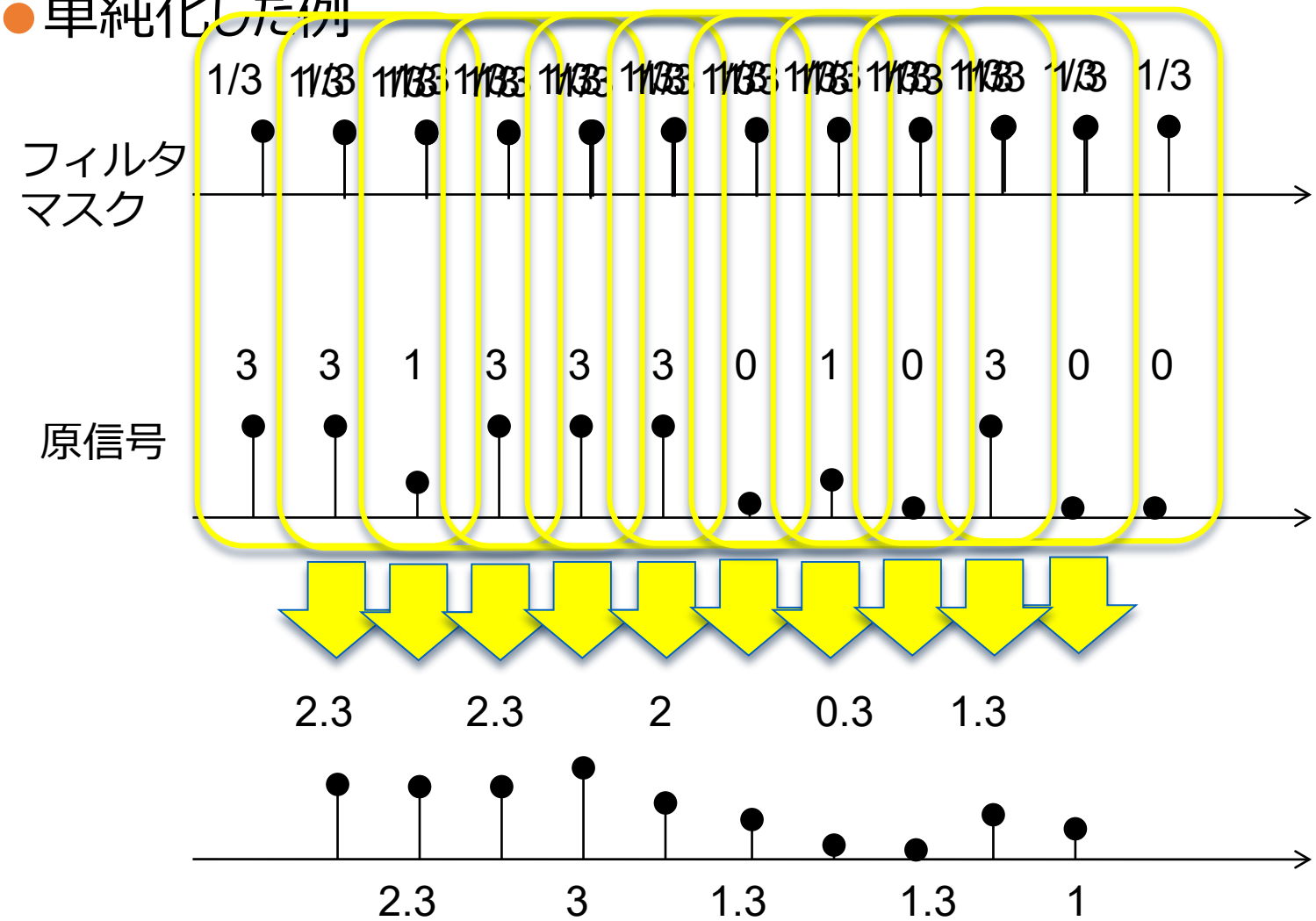


フィルタリング



最も基本的なフィルタ：平滑化

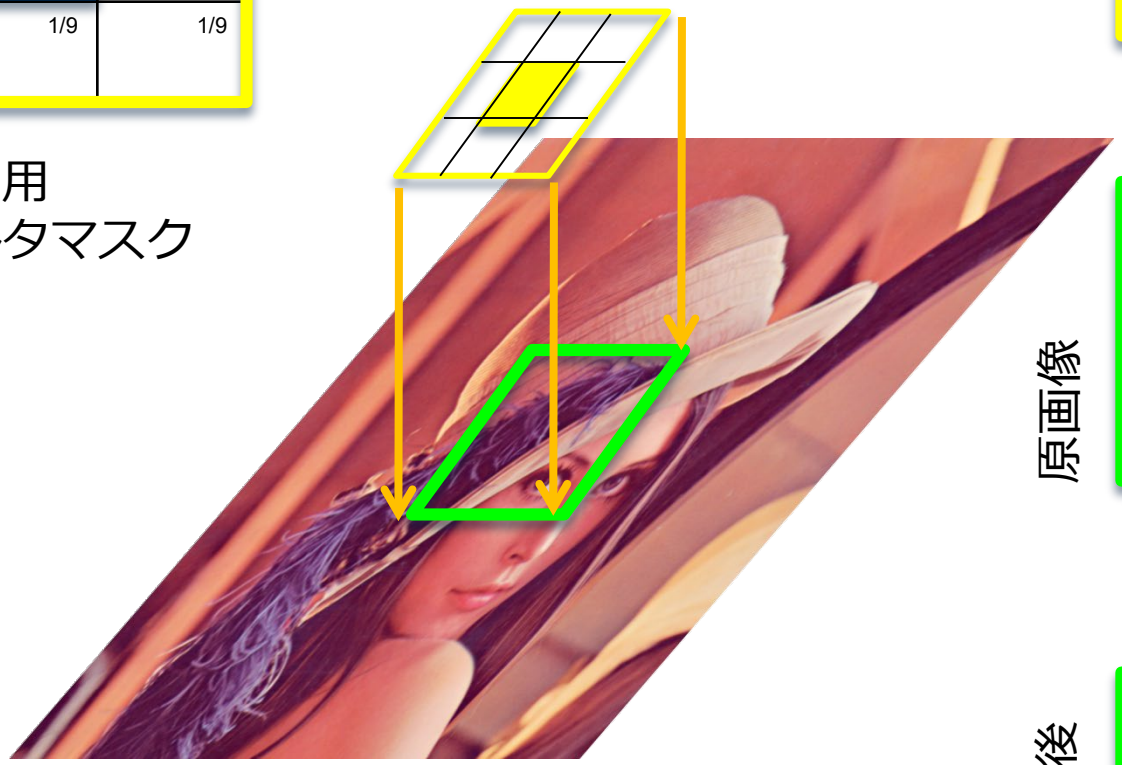
- 単純化した例



最も基本的なフィルタ：平滑化

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

平滑化用
フィルタマスク
(3x3)



マスク

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

内積

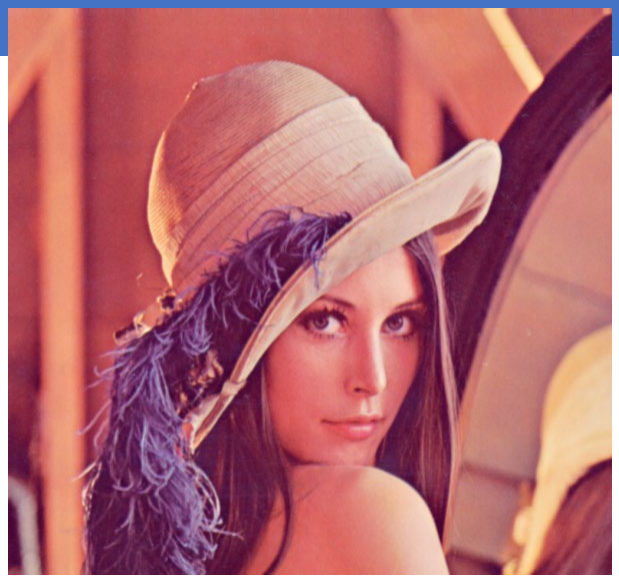
原画像

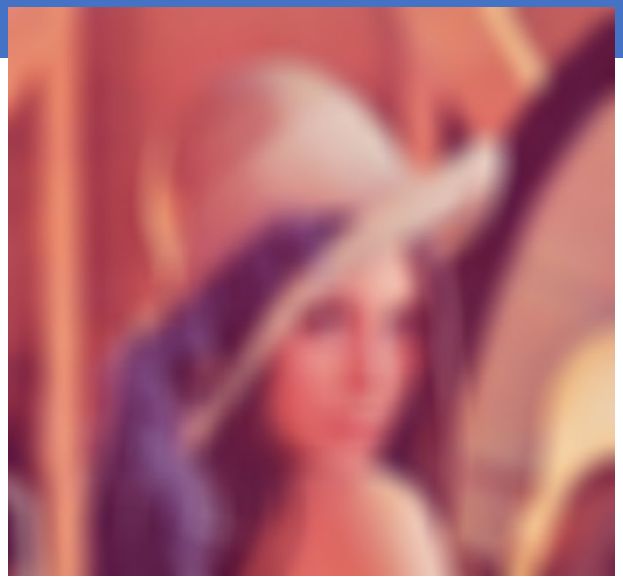
10	10	10
10	20	10
10	10	10

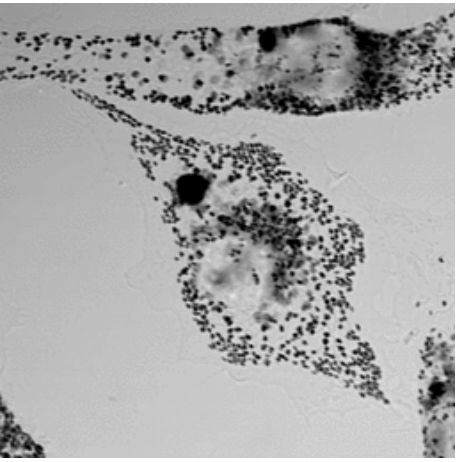
平滑化後

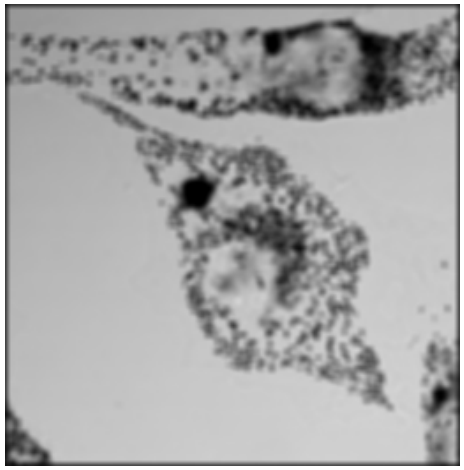
10	10	10
10	11	10
10	10	10

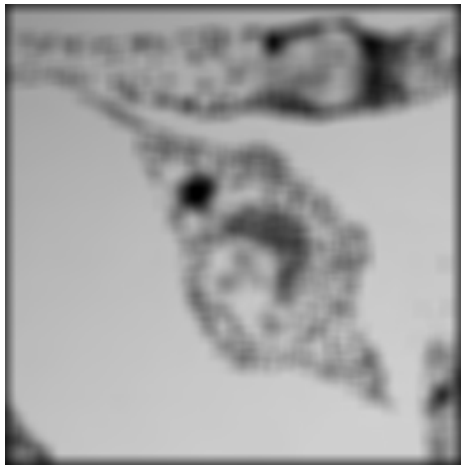
最も基本的なフィルタ：平滑化











平滑化用のフィルタマスク

0	0	0
0	1	0
0	0	0

無処理のオペレータ
（そのまま出力）

0.11	0.11	0.11
0.11	0.12	0.11
0.11	0.11	0.11

平均を求める
オペレータ
（平滑化）

0.05	0.05	0.05
0.05	0.60	0.05
0.05	0.05	0.05

中心重点の平均を
求めるオペレータ
（ぼけ過ぎない平滑化）

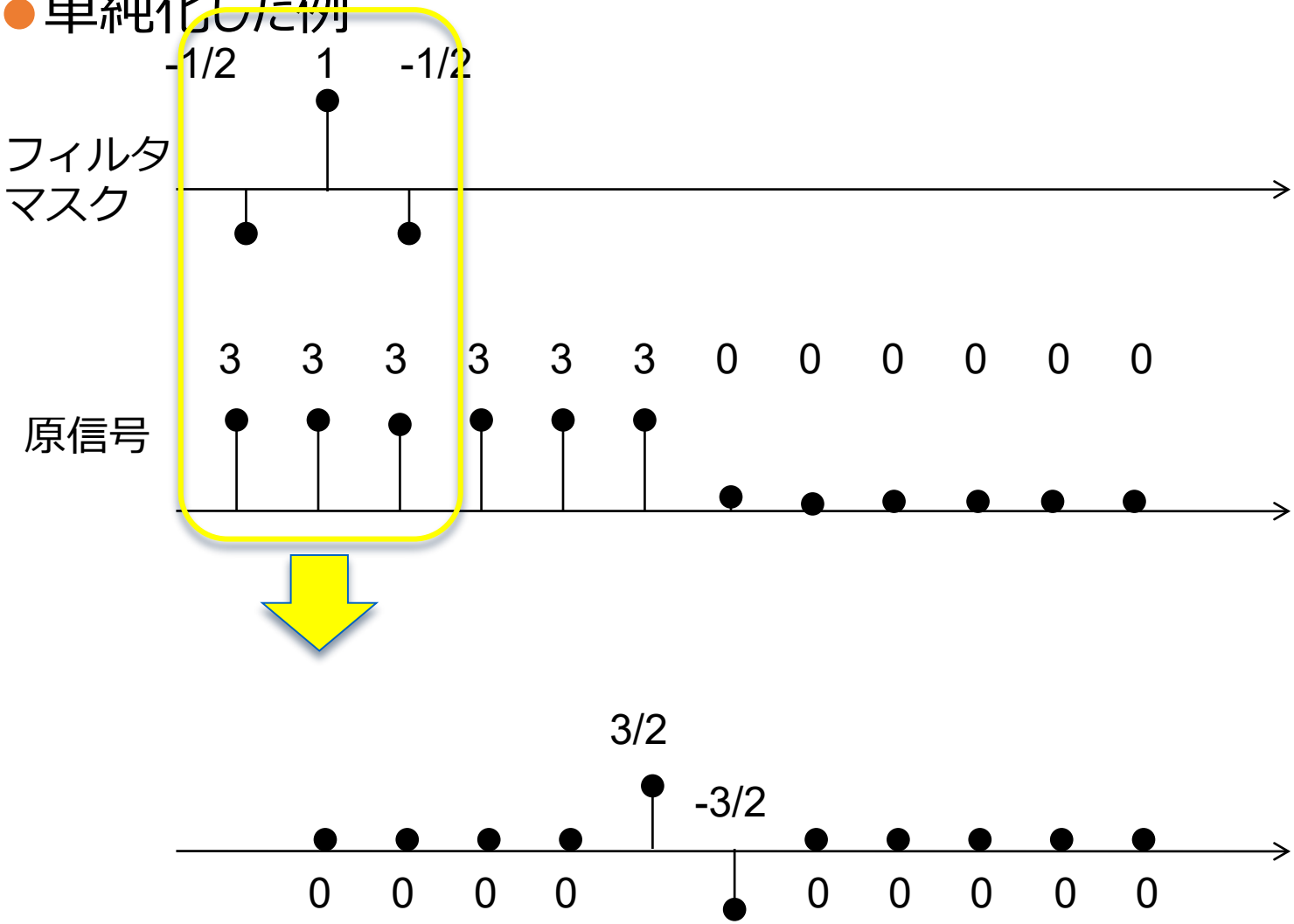
http://codezine.jp/article/detail/129

7.1 フィルタリングとは

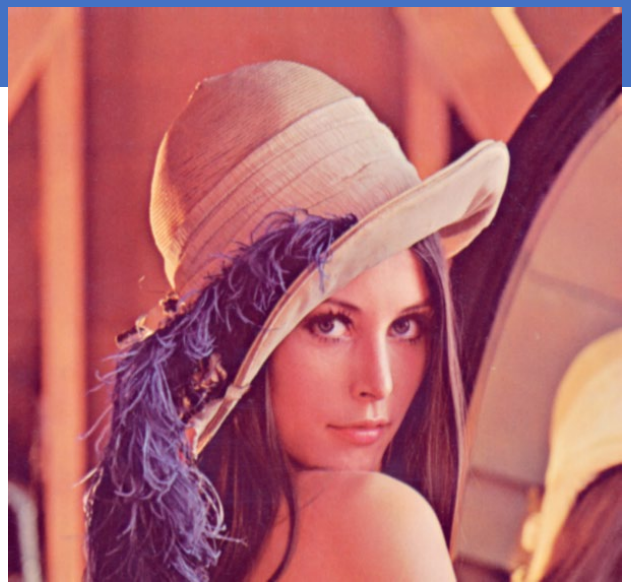
- 画像上の各画素を中心とした局所領域で何らかの計算
- 画質改善や特徴抽出を目的

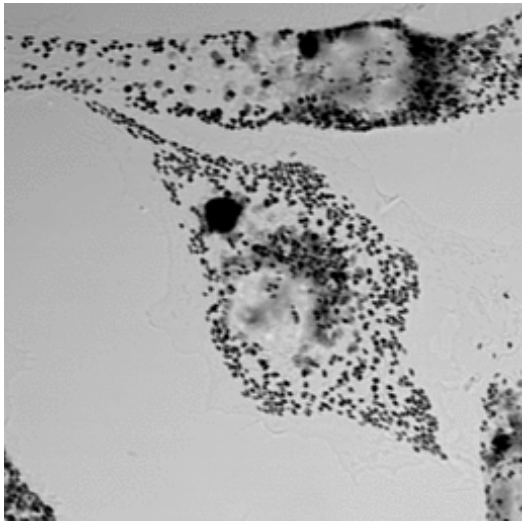
基本的なフィルタ：エッジ検出

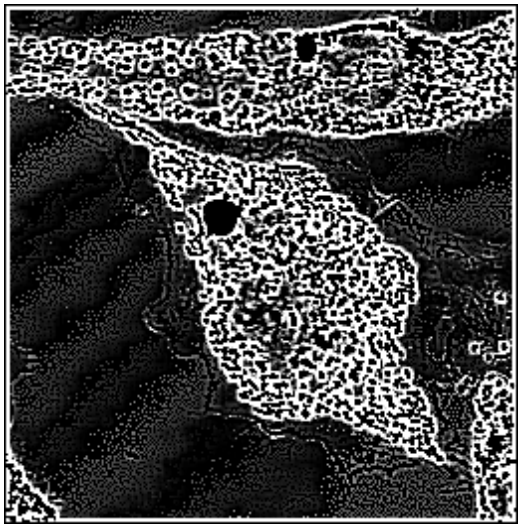
- 単純化した例



基本的なフィルタ：エッジ検出



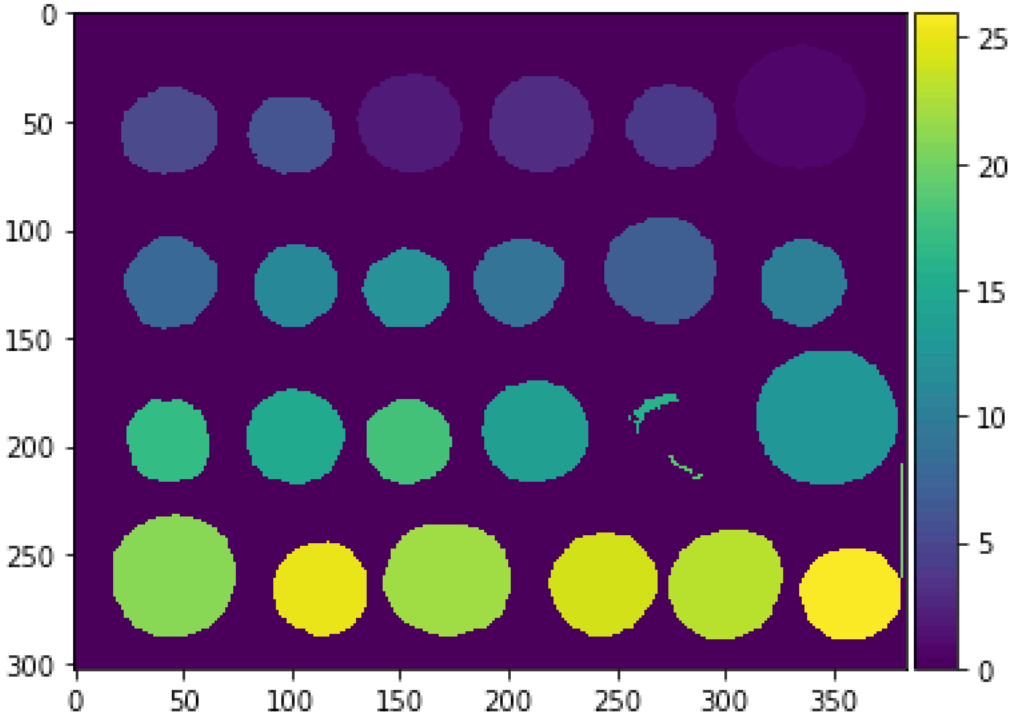




Painting images with labels

- 一まとまりの領域ごとにID(label)をつける
- ndi.label

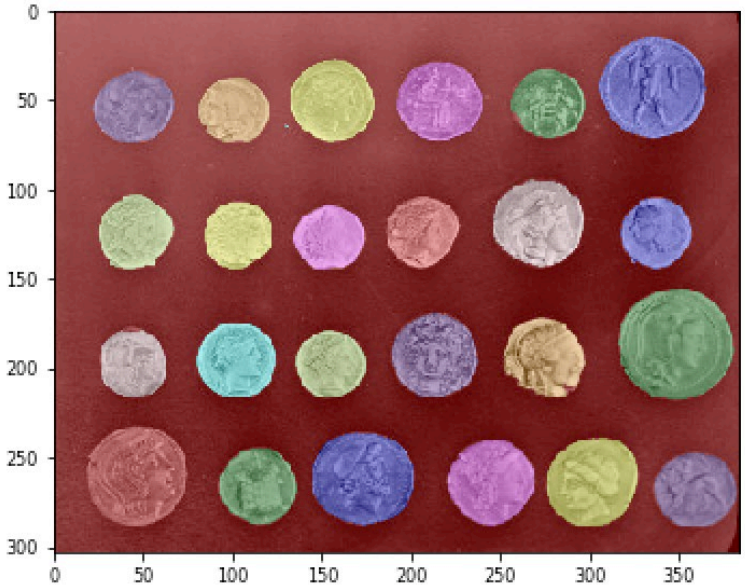
```
labeled_coins, _ = ndi.label(coins_cleaned) # Label all coins one by one
io.imshow(labeled_coins)
```



Visualize results (1)

- label2rgb() でラベルごとに独立した色を与えて可視化.

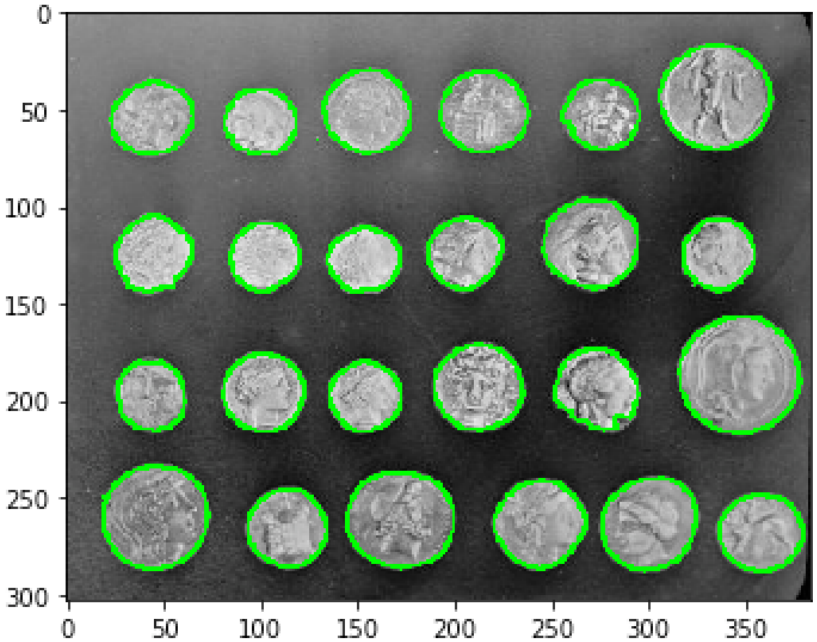
```
from skimage.color import label2rgb
image_label_overlay = label2rgb(labeled_coins, image=coins)
io.imshow(image_label_overlay)
```



Visualize results (2)

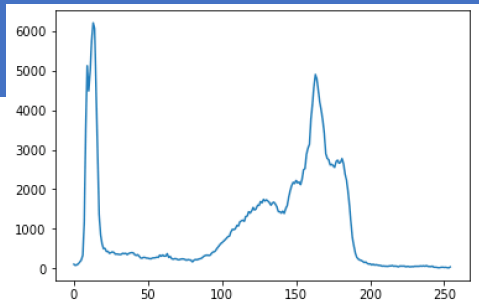
- オブジェクトの境界を可視化
  - skimage.morphology.erosion

```
contours = segmentation ^ ndi.binary_erosion(segmentation, np.ones((5,5)))

from skimage.color import gray2rgb
coins_rgb = gray2rgb(coins)
coins_rgb[contours] = [0,255,0]
io.imshow(coins_rgb)
```

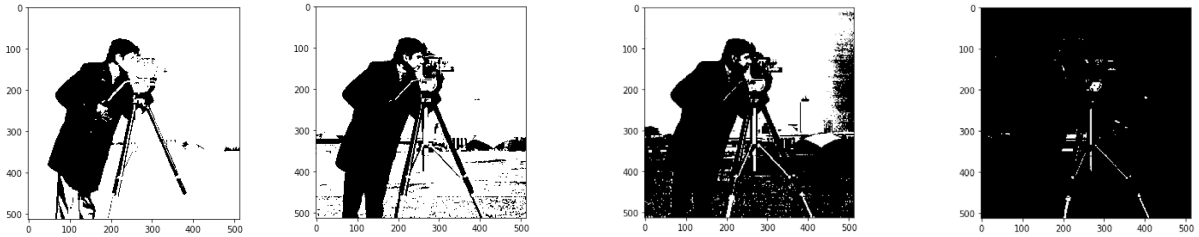


演習

演習 1 (thresholding)



- ”camera”画像の呼び出し
- 輝度値のヒストグラム可視化
- 複数の閾値を用いて、セグメンテーションの結果を比べよう！

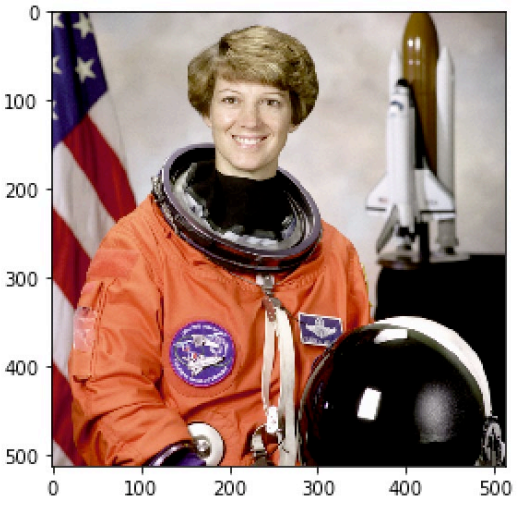


- ヒント:
  - skimage.data.camera()
  - numpy.histogram()
  - 画像配列をベクトルに直す方法

    image.reshape(-1)

演習 2

- “astronaut” 画像を読み込み、RGB成分それぞれの画像を表示してみよう



- ヒント:
  - image = data.astronaut()
  - image.shape: (512L, 512L, 3L)

演習 3 (image stiching)

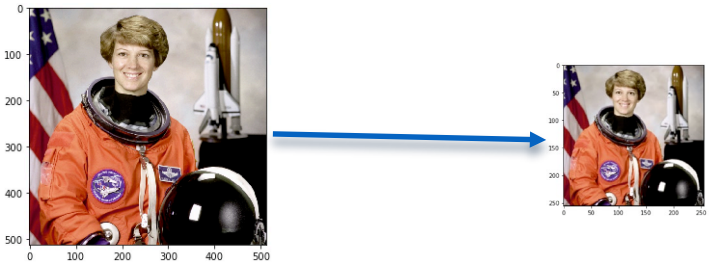
- ４つの画像(im1.tiff, im2.tiff, im3.tiff, im4.tiff)を読み込んで、つなげることで一つの画像を作って、保存しよう！

- ヒント:
  - 画像の読み込み： io.imread
  - 画像の大きさ：shape attribute
  - 画像の初期化： numpy.zero (specify np.uint8 for the type to show nice results)
  - 画像の保存 io.imsave

演習 4 (downsampling)

- “atronaut” 画像を読み込み, 画像を2分の１のサイズにダウンサンプリングせよ
  - 関数を利用する
- ヒント
  - from skimage.transform import downscale_local_mean
  - downscale_local_mean(image, (2,2,1))

    → 2×2×1のウィンドウで平均値を求めてその値を利用



演習 5

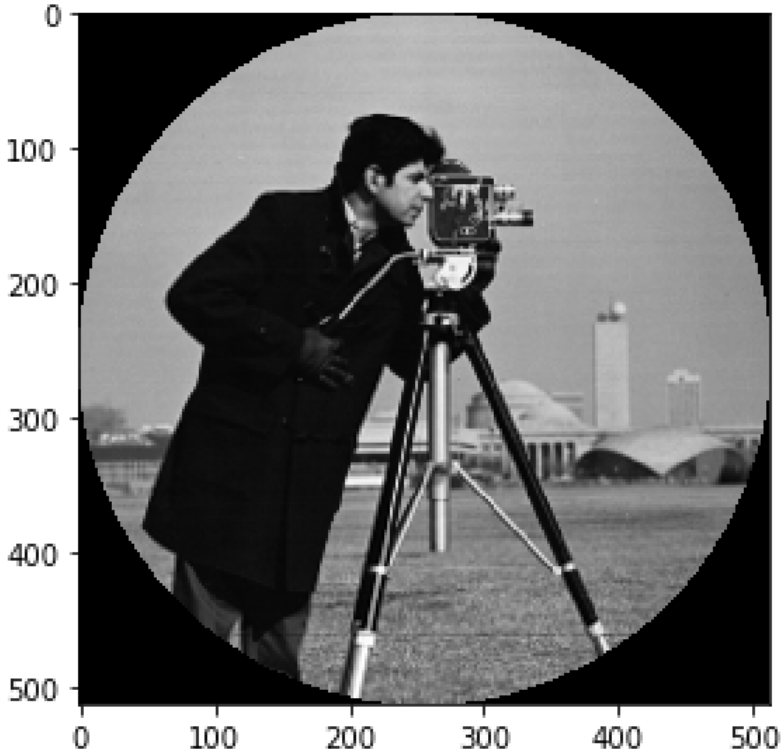
- “coins”画像を読み込み、色んなコントラスト強調を試してみよう

- ヒント:
  - from skimage import data, io
  - coins = data.coins()
  - 36ページ

Advance
早く終わった人はやってみよう

演習 6

- “camera"画像を読み込み、下図のように円状（中心：画像の中心、半径R：画像の縦横のサイズの半分）の外の画素値を全て０とするプログラム



円の方程式

$$\{(i,j)|(i-c_i)^2+\left(j-c_j\right)^2<R^2)\},$$

※)

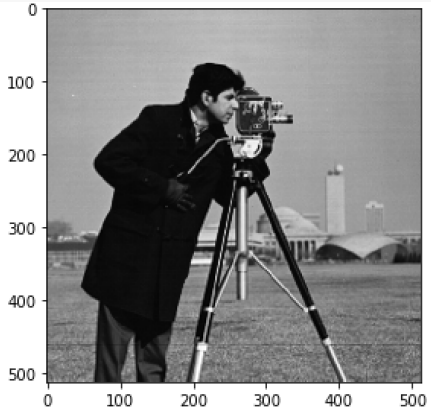
(i,j) ：ピクセルの座標

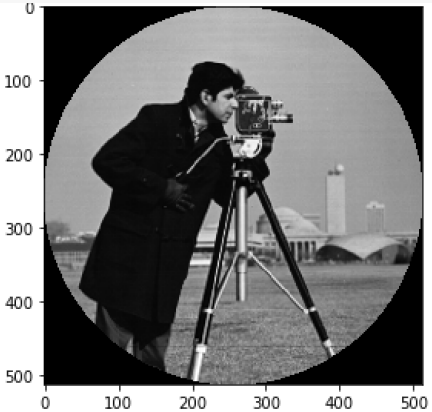
(c_i,c_j) ：円の中心座標

R ：円の半径

Answer

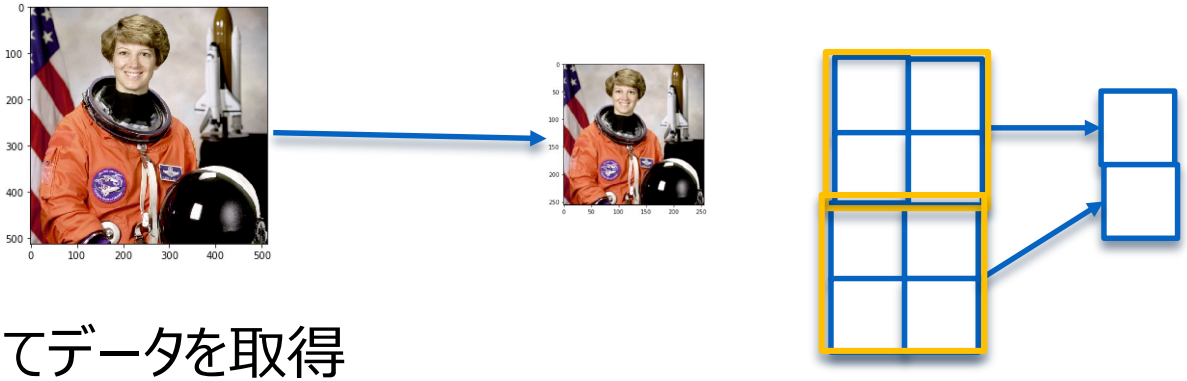
```
camera = data.camera() # load the image camera
io.imshow(camera)   # Display the image
io.show()

import numpy as np
nrows, ncols = camera.shape # Get the dimension of the image
                            # (number of rows and number of columns)
cnt_row, cnt_col = nrows / 2, ncols / 2 # Center of the disk

camera2 = camera.copy()
for row in range(nrows): # 画像の縦方向の繰り返し
    for col in range(ncols): # 画像の横方向の繰り返し
        # 各ピクセルについて、画像の中心からの距離が閾値R（円の半径）より離れていれば、フラグを立てる。
        outer_flag = ((row - cnt_row)**2 + (col - cnt_col)**2 > (nrows / 2)**2)
        if outer_flag:# 上の条件を満たせば、
            camera2[row,col] = 0 # ピクセル値を0（黒）にする

io.imshow(camera2)   # Display the image
io.show()
```





演習 4 : Advance (downsampling)

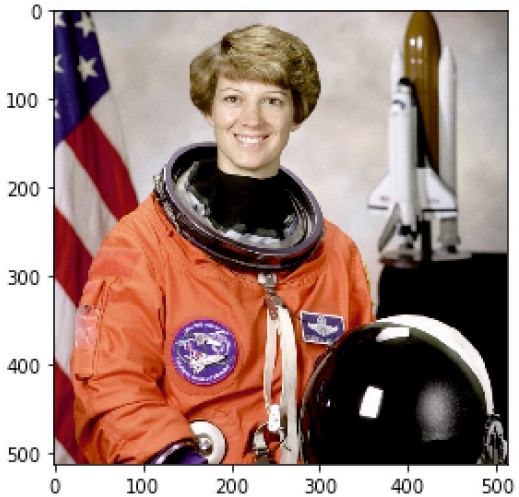
- “atronaut” 画像を読み込み, 画像を2分の１のサイズにダウンサンプリングしよ
  - 周辺の画素の平均値を求めて、新画像の1画素の値にする



- ヒント
  - float型に変換してデータを取得
    - image = data.astronaut().astype(float)
  - 行と列の数の取得
    - nrows, ncols, _ = image.shape (color image は 3次元)
  - unsigned int8 型に戻す
    - io.imshow(result.astype(np.uint8))

演習 7

- “atronaut” 画像を読み込み、円状（中心：画像の中心、半径R：画像の縦横のサイズの半分）の外の画素値を全て０とするプログラム



- ヒント:
  - 26ページのカラーバージョン

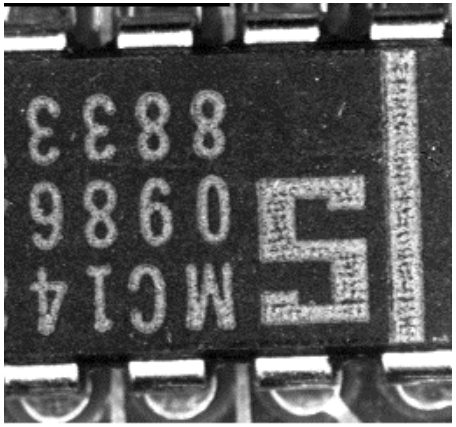
演習 8

- “LENNA.bmp” を読み込み、sobel filter でエッジ抽出してみよう

- ヒント:
  - 60ページ

演習 9

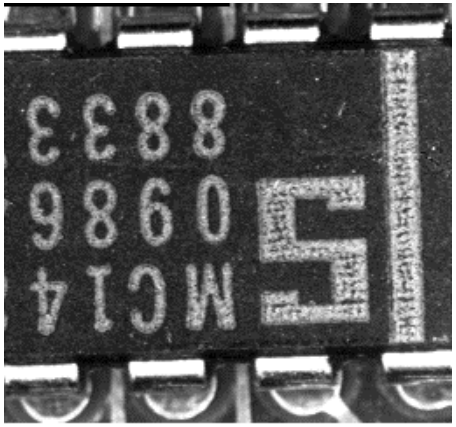
- “Text.bmp”を読み込み、テキスト領域を閾値を用いてセグメンテーションせよ
- 領域ごとにラベリングを行って、領域ごとに色をつけよ



- ヒント:
  - 平滑化してから、輝度値で閾値処理
  - 他の領域を含んでいてもO.K.

演習 10

- “Text.bmp”を読み込み、テキスト領域をwatershedを用いてセグメンテーション



- ヒント:
  - Sobel filter を用いてelevation_mapを作成
  - マーカー領域は、閾値でラフ（明らかに輝度が高い領域を抽出）に抽出
  - from skimage.morphology import watershed
  - 他の領域を含んでいてもO.K.