コンピュータビジョン
Python基礎とOpenCVを使った画像の操作

Pythonを始める

対話型のinterpreter

```
$ python
Python 3.5.2 ... （略）
>>> a = 0.0
>>> for i in range(10):
...     a = a + pow(1.4, i)
...
>>> a
69.81366374399997
```

プログラム型のScript

ファイルの中身：

geomsum.py

```
a = 0.0
for i in range(10):
    a = a + pow(1.4,i)
print(a)
```

シェルからの実行：

```
$ python geomsum.py
69.813663744
```

制御構造

インデントされた部分が一つのブロックとなる

blocks.py

```
x = 0
for i in range(100):
    x = x + i

y = i = 0
while i < 100:
    y = y + i
    i = i + 1

print(x, y)
```

実行結果

```
$ python blocks.py
4950 4950
```

list

```
>>> L=[0,1,2,3,4,5,6,7,8,9]     # same as L = list(range(0,10))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> L[0]        # L[index]
0
>>> L[3]
3
>>> L[1:6]      # L[start:stop]
[1, 2, 3, 4, 5]
>>> L[1:6:2]    # L[start:stop:step]
[1, 3, 5]
>>> L[1:6] = [‘a’,‘ab’,‘abc’,‘abcd’,‘abcde’]
>>> L
[0, ‘a’, ‘ab’, ‘abc’, ‘abcd’, ‘abcde’, 6, 7, 8, 9]
>>> L=[[2,0,3,0],[0,4],[1,5]]
>>> L
[[2, 0, 3, 0], [0, 4], [1, 5]]
```

list (続き)

```
>>> L=list(range(10))
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> X=L
>>> Y=list(L)    #
>>> X
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> L[1]=2
>>> L
[0, 2, 2, 3, 4, 5, 6, 7, 8, 9]
>>> X
[0, 2, 2, 3, 4, 5, 6, 7, 8, 9]
>>> Y
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> X.append(10) # L.insert(index, item) is also available
>>> L
[0, 2, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> Y
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

string（文字列）

```
>>> a = 'This is '
>>> b = 'a pen'
>>> c = a+b
>>> c
'This is a pen’
>>> c[0]
‘T’
>>> c[1]
‘h’
>>> c[5:9]
‘is a’
>>> c[2] = ‘a’
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> d = 'the desk’
>>> print('That is %s.' % d)
That is the desk.
>>> print('The number of %s = %d' % ('pens’,3))
The number of pens = 3
```

関数

関数定義の仕方：

func.py

```
import numpy

def func1(a, b, c):
    return (-b+numpy.sqrt(b*b-4.0*a*c))/(2.0*a)

def func2(a, b, c):
    return (2.0*c)/(-b-numpy.sqrt(b*b-4.0*a*c))

[a,b,c] = numpy.array([1e-6,1,1e-5])
print(func1(a,b,c))
print(func2(a,b,c))
```

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

$$\frac{2c}{-b-\sqrt{b^2-4ac}}$$

実行結果

```
$ python func.py
-1.00000008274e-05
-1.00000000001e-05
```

Module

ファイル名（xxxx.py）を使うとインポート可能

- 追加した赤い行内のブロック：インポート時には実行されない

```
>>> import func2
>>> func2.func2(1,2,-3)
1.0
>>> import func
1.00000008274e-05
1.00000000001e-05
```

func2.py

```
...
if __name__ == '__main__':
    [a,b,c] = numpy.array([1e-6,1,1e-5])
    print func1(a,b,c)
    print func2(a,b,c)
```

Further reading: The Python Tutorial: https://docs.python.org/3/tutorial/

NumPyを始める

数値計算を効率的に行うための拡張モジュール

- まず最初にインポート

```
>>> import numpy
```

- インポートのバリエーション
  - 通常 numpy.X のように書くが，np.X と書けるようにする：

```
>>> import numpy as np
```

  - さらに省略して X と書けるようにする：

```
>>> from numpy import *
```

  - 参考）namespace

NumPy：Arrayの生成1

- NumPyはベクトル・行列・テンソルを扱える
- list からの生成
  - ベクトル

```
>>> a = numpy.array([2,0,1,4], float)
>>> a
array([ 2.,  0.,  1.,  4.])
```

  - 行列

```
>>> a = numpy.array([[2,0],[1,4]], int)
>>> a
array([[2, 0],
       [1, 4]])
```

  - テンソル

```
>>> a = numpy.array([[[2,1],[0,2],[1,-1]],[[1,3],[3,4],[3,5]]])
```

NumPy：Arrayの生成2

- 便利な生成関数
  - 零行列

```
>>> numpy.zeros((2,4))
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

  - 全要素が1

```
>>> numpy.ones((3,1))
array([[ 1.],
       [ 1.],
       [ 1.]])
```

  - 単位行列

```
>>> numpy.eye(3)
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

NumPy：Array要素へのアクセス

- 単一要素：添字は0,1,…

```
>>> a = numpy.array([[2,0,1],[1,4,5]], float)
>>> a
array([[ 2.,  0.,  1.],
       [ 1.,  4.,  5.]])
>>> a[0,2]
1.0
```

- 列ベクトル・行ベクトル

```
>>> a[:,1]
array([ 0.,  4.])
>>> a[1,:]
array([ 1.,  4.,  5.])
```

- 部分行列

```
>>> a[:,1:3] # the same as a[:,1:] in this case
array([[ 0.,  1.],
       [ 4.,  5.]])
```

NumPy：その他

- ランダムな要素を持つ行列を生成

```
>>> a=numpy.random.rand(3,4)
>>> a
array([[ 0.64569868,  0.35669136,  0.34230599,  0.00687083],
       [ 0.29524555,  0.06356345,  0.482582  ,  0.09736972],
       [ 0.38808636,  0.10640061,  0.54865189,  0.13813626]])
```

- サイズを知る

```
>>> a.shape
(3, 4)
```

- 型を知る
  - uint8,uint16,int8,int16,float32,float64 などがある

```
>>> a.dtype
dtype('float64')
```

NumPy：コピー

- = は複製を作らない
  - 関数に渡した変数も複製されない
- 別のArrayにコピーするには copy() を使用
  - deep copy と呼ばれる

```
>>> a=numpy.array([[2.,0.],[1.,4.]])
>>> a
array([[ 2.,  0.],
       [ 1.,  4.]])
>>> b=a
>>> a[1,1]=3
>>> a
array([[ 2.,  0.],
       [ 1.,  3.]])
>>> b
array([[ 2.,  0.],
       [ 1.,  3.]])
```

```
>>> c=b.copy()
>>> c
array([[ 2.,  0.],
       [ 1.,  3.]])
>>> c[1,1]=5
>>> c
array([[ 2.,  0.],
       [ 1.,  5.]])
>>> a
array([[ 2.,  0.],
       [ 1.,  3.]])
```

OpenCVの画像とNumPy.Array

- OpenCVで扱う画像は，NumPyのArrayである
  - width × height → (height, width)
  - カラー画像

```
>>> image = numpy.zeros((480, 640, 3), numpy.uint8)
>>> image.shape
(480, 640,  3)
>>> lion = cv2.imread('../1st/lion.jpg')
>>> lion.shape
(540, 960,  3)
```

  - グレースケール画像

```
>>> lion = cv2.imread('../1st/lion.jpg', 0)
>>> lion.shape
(540, 960)
```

矩形の描画

mondorian.py

```
import cv2
import numpy
import random

image = numpy.zeros((480, 640, 3), numpy.uint8)

while 1:
    color = (int(random.random()*255),¥
             int(random.random()*255),¥
             int(random.random()*255))

    pt1 = (int(random.random()*640), int(random.random()*480))
    pt2 = (int(random.random()*640), int(random.random()*480))

    cv2.rectangle(image, pt1, pt2, color, -1)

    cv2.imshow('Mondrian', image)
    key = cv2.waitKey(100)

    if key == 0x1b: # ESC
        break
```

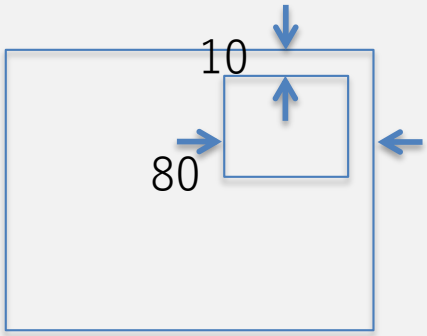
課題

- 色を同じ特定の色にするには？
- 四角を正方形にするには？

画像の重ね合わせ

superimpose.py

```
import cv2
import numpy

cap = cv2.VideoCapture(0) # or ('1st/sample.avi')
logo = cv2.imread('cv_logo.png')
logo_mask = (1-numpy.minimum(logo, 1))*255

while 1:
    stat, image = cap.read()

    w = image.shape[1]
    roi = image[10:10+logo.shape[0], w-80:w-80+logo.shape[1], :]
    image[10:10+logo.shape[0], w-80:w-80+logo.shape[1], :]¥
         = (roi & logo_mask) | logo

    cv2.imshow('Video', image)
    key = cv2.waitKey(10)

    if key == 0x1b: # ESC
        break
```



課題

- ロゴを出す位置を左下にするには？
- “(roi & logo_mask) | logo”を
  - logo
  - roi ^ logo
- にしてみると？

画像のヒストグラム

color_hist.py

```
...
while 1:
    stat, image = cap.read()

    nbins = 64
    hist = numpy.histogram(image[:,:,1].ravel(), nbins)
    scale = nbins/float(image.shape[0]*image.shape[1])*30

    pts = numpy.zeros((nbins, 2), numpy.int32)
    for i in range(nbins):
        pts[i,:] = numpy.array([hist[1][i]*2,¥
                                300 - float(hist[0][i])*scale])

    cv2.polylines(image, [pts], False, (0,255,0))

    cv2.imshow('Video', image)
    key = cv2.waitKey(10)

    if key == 0x1b: # ESC
        break
```

イメージセンサのノイズ

image_noise.py

```
import cv2
import numpy

cap = cv2.VideoCapture(0)
x0, y0 = 320, 240
n_samples, cnt, nbins, log = 300, 0, 12, 0
brtness = numpy.zeros((n_samples, 3), numpy.uint8)

while 1:
    stat, image = cap.read()

    if log == 1:
        brtness[cnt,:] = image[y0,x0,:]
        cnt = cnt + 1
    else:
        print image[y0,x0,:]
    if cnt == n_samples:
        break

    cv2.line(image, (x0+10,y0), (x0-10,y0), (0,255,0))
    cv2.line(image, (x0,y0+10), (x0,y0-10), (0,255,0))
```

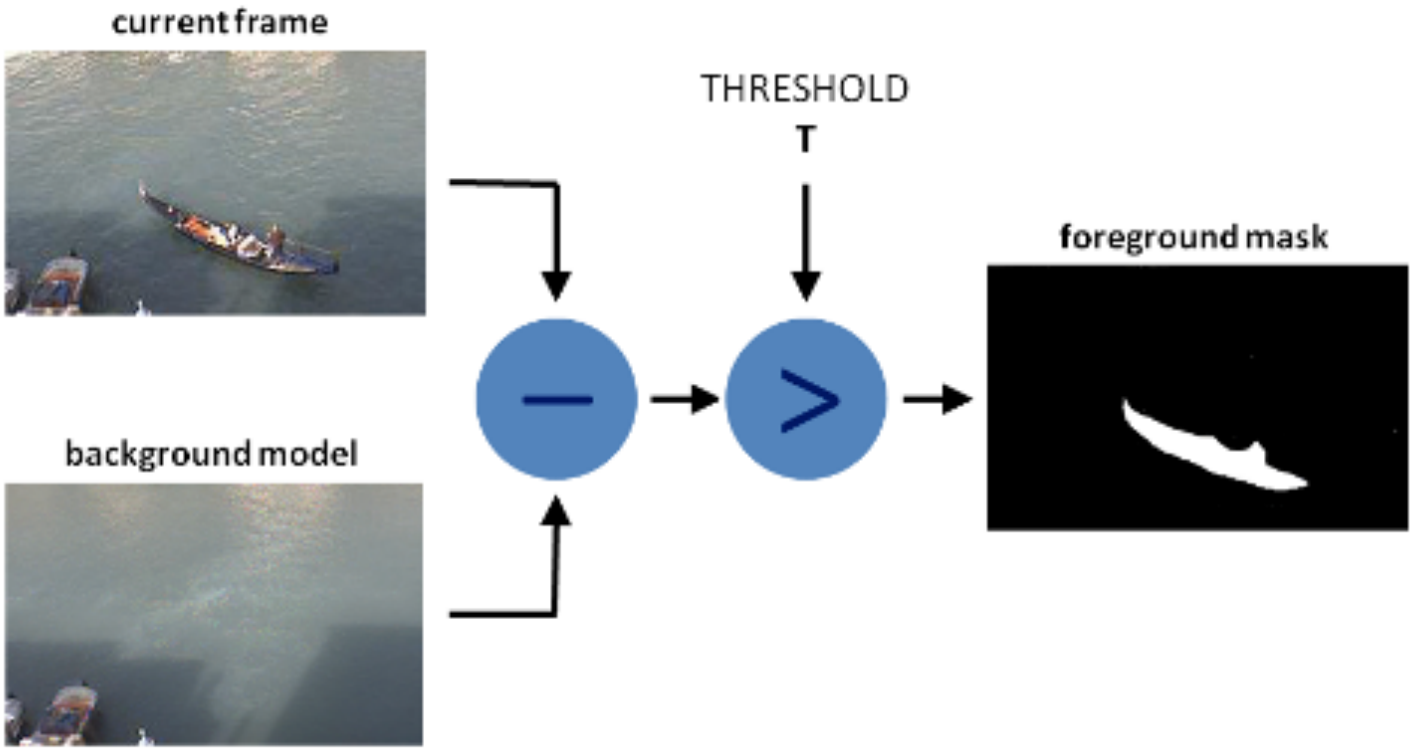
イメージセンサのノイズ

image_noise.py(contd.)

```
    cv2.imshow('Camera', image)
    if cv2.waitKey(10) == 0x1b:
        print 'start!'
        log = 1

channel = 0
br_mean = numpy.mean(brtness[:,channel])
br_std = numpy.std(brtness[:,channel])
print 'brightness = %f +- %f' % (br_mean, br_std)
histo = numpy.histogram(brtness[:,channel], nbins,¥
                  (int(br_mean)-nbins/2, int(br_mean)+nbins/2))

for i in range(len(histo[0])):
    print histo[1][i], histo[0][i]

cap.release()
cv2.destroyAllWindows()
```

Background subtraction

- 背景差分：最も基本的なシーン変化検出・物体追跡手法
  - 現在のフレームと基準となるフレームの差分を計算
  - 基準となるフレームとは？（ノイズやシーン変化の存在）



http://docs.opencv.org/trunk/doc/tutorials/video/background_subtraction/background_subtraction.html

Background subtraction

判定しようとする画素のRGB値

$$R = \frac{p(BG|\mathbf{x})}{p(FG|\mathbf{x})}$$

判定

$R > 0.5$ → Background

$R < 0.5$ → Foreground

Bayes' rule

$$= \frac{p(\mathbf{x}|BG)p(BG)}{p(\mathbf{x}|FG)p(FG)}$$

背景と前景の出やすさは不明：

$$p(BG) = p(FG) = \text{const.}$$

前景画素の見え方は不明：

$$p(\mathbf{x}|FG) = \text{const.}$$

判定は次に帰着

$$p(\mathbf{x}|BG) > c_{th}$$

背景の画素のモデル
（ガウス分布・混合ガウス分布）

Background subtraction

bgsubtract.py

```
import cv2

cap = cv2.VideoCapture('vtest.avi')

bgsub = cv2.createBackgroundSubtractorMOG2()

while(1):
    ret, frame = cap.read()

    fgmask = bgsub.apply(frame)
    fgmask_bgr = cv2.cvtColor(fgmask, cv2.COLOR_GRAY2BGR)

    cv2.imshow(’Camera’, image & fgmask_bgr)
    k = cv2.waitKey(10)

    if k == 0x1b:
        break

cap.release()
cv2.destroyAllWindows()
```