Python + OpenCV プログラミングの基礎

Mac OSが手元にないのでWindows用の解説です
Windows 7（MacBookPro bootcamp環境）でのみ動作確認

最終編集日 2017/1/29

インストールする

メモ

- その昔、人はPython本体とライブラリ類（NumPy/SciPy/MatplotLib）を一つずつインストールしてた
  参考http://qiita.com/sky_y/items/4b9641b01e713ea4ab73
- 現在、全部一度にインストールしてくれるAnacondaというのがある
  参考：http://qiita.com/y__sama/items/5b62d31cb7e6ed50f02c
  本家：https://www.continuum.io/downloads
  今回はこれを利用
- OpenCVもAnacondaからインストールできます

Anacondaをインストールする

- 本家ページより，インストーラーをダウンロード
  - https://www.continuum.io/downloads
  - 今回はPython3.5, 64bit installerを選択
  - ファイルサイズ391MBなので、ちょっとだけ時間がかかる
- 『Anaconda3-4.2.0-Windows-x86_64.exe』を起動しインストール
  - コマンドプロンプトで > python –versionとして以下の感じなら成功

OpenCVをインストール

1. コマンドプロンプトを右クリックして管理者権限で起動
2. > conda install –c menpo opencv3
3. 途中でyキーを押す
4. 5分くらい待つ
5. > python sample.py

```
sample.py
img = cv2.imread('sample.jpg')
print( img.shape )
cv2.imshow('sample', img)
cv2.waitKey(0)
```

引くほど簡単にインストールできた
本当はもっと大変な思いをすると思ったからポイントをメモしておくためにこの文章を書き始めたのに。。。

エディタ

- エディタはなんでも良いと思います（Vim/Emacs/limetext/xzy）
- 私は手元にあったのでVisual Studio 2015 communityを利用

- Python environmentをインストール（20分位かかった）
- 右側にあるPython Environmentにおいて

『+ Custom』をクリックし新しいenvironmentを生成

Confgureタブより以下を指定

Prefix path	: **Program Files(x86)/Anaconda2**
Interpreter Path	: **Program Files(x86)/Anaconda2/python.exe**
Windowed Interpreter	: **Program Files(x86)/Anaconda2/pythonw.exe**
Library path	: **Program Files(x86)/Anaconda2/python.exe**

Intelli senseタブよりrefreshする (時間かかる)

- これでインテリセンスが効くようになる

Pythonの基礎

Pythonの基礎

- デジタルメディア処理で紹介するコードを理解するための最小限必要な文法を解説します
- 主に井尻の勉強のための、ごく基本的な内容の覚え書きです
- 初学者ゆえ間違いがあったり非効率なコードあると思います
  → ご指摘いただけると非常にありがたいです

型

整数型-int-

```
a = 1234        #整数1234
a = Oxffff      #整数 16進数
a = Ob10010     #整数 2進数
```

長整数-long-(python3では廃止)

```
a = 1234567890123L
```

浮動小数点-float-

```
a = 1.234    #浮動小数点
a = 1.2e3    #1.2 * 10^3
```

論理値-bool-

```
a = True
a = False
```

文字列-string-

```
a = 'Hello World'
a = "Hello World"
a = 'Hello ' 'World'
```

型変換 (以下、例を少しだけ)

```
a = int(16.2)     #a=16   (float→ int)
a = int("16")     #a=16   (str  → int)
a = float(16)     #a=16.0(int  →float)
```

型判定

```
print type(a)     #aの型が表示される
```

配列型：tuple/list/np.array (1/3)

(1,2,3)	tuple	：**長さ&値 変更不可**の配列
[1,2,3]	list	：可変長配列
np.array(1,2,3)	np.array:	*n*次元配列（画像などはこれで表現される）

- np.arrayは高速処理のための制約がある配列
  - np.array では要素がメモリ内の連続領域に配置される
  - np.array では各次元の要素数は等しい（行列の形になる）
  - np.array では原則的に要素は同じ型
  - 参照 → http://www.kamishima.net/mlmpyja/nbayes1/ndarray.html

```
# python のlistもlinked listではなく配列で実装されているっぽい
# ただ多重になったときは連続領域を取るのは難しそう
# http://stackoverflow.com/questions/3917574/how-is-pythons-list-implemented
```

配列型：tuple/list/np.array (2/3)

list1 初期化

```
A = [1,3,5,7]              #list初期化
A = []                     #空のlistを生成
A = [0] * 5                #初期化   [0,0,0,0,0]
A = [0 for i in range(5)]  #内包表記 [0,0,0,0,0]
A = [i  for i in range(5)]  #内包表記 [0,1,2,3,4]
```

#内包表記は最初見たとき驚いたけど便利

list2 操作

```
A = [1,2,3,4]
print( len(A) )    #Aの長さを取得
print( A[2] )      #2番目の要素にアクセス
A.append(4)        #後ろに『4』を挿入
a = A.pop(2)       #2番目の要素をpop
A.remove(4)        #値『4』の最初の要素を削除
```

tuple

```
A =  (1,3,5,7)     #tupleの初期化
print( A[2] )      #2番目の要素にアクセス
A[2] = 3           #エラー tupleは変更不可
```

tuple ←→ list

```
A = (1,2,3,4)      #tuple
B = list(A)        #tupleからlistへ変換
C = tuple(B)       #listからtupleへ変換
```

型変換 map()

```
A = [1,2,3,4]
B = map(float, A) #全要素をfloat型に変換
C = map(str, A)   #全要素を文字列型に変換
```

配列型：tuple/list/np.array (3/3)

Np.array 初期化

```
A = np.array([1,2,3,4])    #listで初期化
B = np.array((1,2,3,4))    #tupleで初期化
C = np.zeros(3)            #要素数指定, 要素は0
D = np.ones(3)             #要素数指定, 要素は1
E = np.ones((2,3))         #[[1,1,1][1,1,1]]
F = np.zeros_like(E)       #Eと同じサイズの0配列
F = np.identity(3)         #3x3 単位行列
```

Np.array 参照

```
A = np.array([1.,2.,3.],[4.,5.,6.])
print( A[1][2] )    #要素(1,2)を参照
print( A[1, 2] )    #要素(1,2)を参照
print( A.shape )    #Aのサイズを表すtuple：(2,3)
print( A.ndim )     #Aの次元数：2
print( A.dtype )    #Aの要素の型：float
```

np.array 四則演算

```
A = np.array([1,2,3,4])    #listで初期化
B = np.array((1,2,3,4))    #tupleで初期化

C = A+B      #要素ごとの和
D = A-B      #要素ごとの差
E = A*B      #要素ごとの積
F = A/B      #要素ごとの商
G = A%B      #要素ごとの余
H = A**B     #要素ごとのべき乗
I = A&B      #要素ごとのAND
J = A|B      #要素ごとのOR
K = A^C      #要素ごとのXOR
```

関数とループ

関数定義

```
#関数は, 次の形で定義する
#複数変数を返り値として返せる
#インデント省略不可

def  関数名(引数1, 引数2) :
      処理1
      処理2
      return a,b
```

For文

```
#i = 0,1,2,3,4 をループする
for i in range(0,5) :
      処理1
      処理2

#リストの要素でループする
A = [1,3,5,7]
for i in A :
      処理1
      処理2
```

OpenCV for Pythonの基礎

OpenCV ファイルIO

OpenCVを用いた画像IO

```
import numpy as np
import cv2

#画像をnp.arrayとして読み込む
img = cv2.imread(ファイル名)

#np.arrayを画像として書き出す
cv2.imwrite(ファイル名, npArray )

#windowを作成し画像を表示
cv2.imshow('window名', img )
cv2.waitKey(0) #キー入力待ち（windowが消えないように）
```

カラー画像からRGB画像を取り出す

```
import numpy as np
import cv2

img = cv2.imread('sample.jpg') # load image

# extract rgb images
imgR = np.zeros( (img.shape[0],img.shape[1]), np.uint8 )
imgG = np.zeros( (img.shape[0],img.shape[1]), np.uint8 )
imgB = np.zeros( (img.shape[0],img.shape[1]), np.uint8 )

for v in range(0,img.shape[0]) :
    for u in range(0,img.shape[1]) :
        imgR[v,u] = img[v,u,2]
        imgG[v,u] = img[v,u,1]
        imgB[v,u] = img[v,u,0]

#export image
cv2.imwrite("R.jpg", imgR)
cv2.imwrite("G.jpg", imgG)
cv2.imwrite("B.jpg", imgB)
```

随時追加していきます…