デジタルメディア処理2

担当: 井尻 敬

デジタルメディア処理２、2017（前期）

~~4/13 デジタル画像とは ：イントロダクション~~
~~4/20 フィルタ処理1 ：画素ごとの濃淡変換、線形フィルタ，非線形フィルタ~~
~~4/27 フィルタ処理2 ：フーリエ変換，ローパスフィルタ，ハイパスフィルタ~~
~~5/11 画像の幾何変換１：アファイン変換~~
~~5/18 画像の幾何変換２：画像の補間，イメージモザイキング~~
~~5/25 画像領域分割 ：領域拡張法，動的輪郭モデル，グラフカット法，~~
~~6/01 前半のまとめ (約30分)と中間試験（約70分）~~
~~6/08 特徴検出1 ：テンプレートマッチング、コーナー・エッジ検出~~
~~6/15 特徴検出2 ：DoG、SIFT特徴量、Hough変換~~
~~6/22 画像認識1 ：パターン認識概論，サポートベクタマシン~~
~~6/29 画像認識2 ：ニューラルネットワーク、深層学習~~
~~7/06 画像処理演習 ：ImageJを用いた画像処理入門~~
7/13 画像処理演習 ：Pythonを用いた画像処理プログラミング入門
7/20 **後半のまとめ (約30分)と期末試験（約70分）**

目的

- 画像処理プログラミングがどのようなものか体験してみる

注：本講義で取り扱うのはあくまでほんの触りの部分だけです．もし興味が湧いた方は，後期の高度情報処理演習Bに来るか，独学で学修を進めてください．

注：本講義では，コードを書きながらPythonの表面的な使い方を体験します．網羅的な機能・文法の紹介は行ないません．

準備：好きな画像を用意しimg.pngという名前で保存してください

Python

- 最近流行りのスクリプト言語
- 機械学習関連のライブラリが充実
- 画像処理関連のライブラリも充実（OpenCV）
- 開発コストが低い（井尻が普段利用しているC++に比べて）

- コードの可読性が高い
- インデントでブロックを強制
  → 変なコードが生成されにくく，学生のコードを読む側としてはとてもありがたい．

自分のPCでpythonを動かしたい人向けメモ

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

  ※2017/3/9現在，Anaconda4.3.0が最新だが，このバージョンはOpenCVがうまくインストールできない

  ※ https://repo.continuum.io/archive/index.html ←こちらから「 Anaconda2-4.2.0-Windows-x86_64.exe 」を利用するとうまくいく．

- 『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

```
C:¥Users¥takashi>python --version
Python 3.5.2 :: Anaconda 4.2.0 (64-bit)

C:¥Users¥takashi>
```

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

```
1 sample.py
2 img = cv2.imread('sample.jpg')
3 print( img.shape )
4 cv2.imshow('sample', img)
5 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（Atom/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

- 2017/3/14追記. 結局Atomに乗り換えました. → https://atom.io/

初めてのPython

Ex1.py “Hello world”

右のコードを動かしてください

1. “ex1.py”というファイルを作成
2. 右のコードを記入
3. コマンドラインで以下を入力
   $ python ex1.py

※ 『#』でコメントアウト
※ 『print(“文字列”)』で文字列を出力

```
# ex1.py

print("hello, world")
```

Ex2.py 変数の型

- int, float, string, boolなどの型を利用可能
- 変数の型は代入する値に応じて自動で決まる
  （型を明示した変数宣言は行わない）
- 後から異なる型に変更することも可能
  （その都度新しい変数が生成される）

右のコードを動かしてみてください

右のコードを色々と編集し型の挙動を確認してください

※ type (変数名) で型を取得
※ id (変数名) でオブジェクトidを取得 → idを見ると，数値代入のたびに新たなオブジェクトが生成されているのが分かる
※ print (変数1, 変数2) で複数変数を出力可能
※ 型変換も可能

```
# ex2.py
#int
a = 1234
print(a, type(a),id(a))

#float
a = 1.234
print(a, type(a),id(a))
a = 1.2345
print(a, type(a),id(a))
a = 1
print(a, type(a),id(a))

#bool
a = True
print(a, type(a),id(a))

#string
a = "hello, world"
print(a, type(a),id(a))

#型変換例: float->int, string->int, int->float
a = int(16.2)
a = int('16')
a = float(16)
```

Ex3.py 配列 (1)

Pythonでは, tuple・list・np.array という配列表現が利用可能

(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
  - 参考：http://www.kamishima.net/mlmpyja/nbayes1/ndarray.html

Ex3.py 配列 (2)

右のコードの出力を予想してください

output1
output2
output3
outout4

コードを実行し，予想と比べてください

コードの中身を色々変化させ，配列とタプルの挙動を確認してください

※ 配列は[], tuppleは()で表現される
※ 配列要素の変更・追加・削除ができる
※ len(配列名)で長さを取得できる
※ 2次元配列（行列）も表現可能

```
# ex3.py

#1D array
A  = [1,3,5,7]
print("output1:", A )

N  = len(A)   #要素数
a2 = A[2]     #2番目の要素を参照
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)   #値が4の最初の要素を削除
print( "output2", N, a, a2, A )

#2D array
A = [[1,2],[3,4],[5,6]]
print( "output3", A[0][1], A, len(A))

#tupple
T = (1,2,3)
# T[1] = 2  #error tappleは変更不可
print("output4",T, T[1])
```

Ex4.py np.array (1)

- np.arrayを含むコードを右に示す

右のコードにprint文を挿入し，計算結果を確認してください

コードの中身を色々変化させ，np.array の挙動を確認してください

※『import numpy as np』はnumpy関連のモジュールのインポート文
※ python & openCV環境では，np.arrayで画像を表現
※ 要素ごとの演算が一行で書ける（画像と画像の和など）
※ np配列名.shapeで配列サイズを取得

```
# ex4.py
import numpy as np

A=np.array([5,6,7,8])
B=np.array([1,2,3,4])
print(A.shape, A)
print(B.shape, B)

#要素ごとの演算(和差積商余べき)
C = A + B
D = A - B
E = A * B
F = A / B
G = A % B
H = A ** B
I = A - 1

#2D
A=np.array([[1,2,3],[4,5,6]])
B=np.array([[1,2,3],[4,5,6]])
C=A+B
print(C,C.shape)
```

Ex5.py np.array (2)

- np.arrayには便利な初期化方法が用意されている
- 要素の総和・平均・分散の計算など，多様な便利機能が用意されている

```
# ex5.py
import numpy as np

A    = np.array([1,2,3,4,5,6,7,8])
mean = np.mean( A )
sum  = np.sum  ( A )
vari = np.var  ( A )
print( mean, sum, vari)

A = A-mean
A = A**2
print( np.sum(A)/A.shape[0]);
```

```
#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
```

Ex6.py for文

- for文は右コードの通り定義できる

右のコードを実行してみてください

右のコードを少し変更して挙動を確かめてください

※インデントによりブロックを定義する（Cでは{}でブロックを定義した）
※ブロック開始部分に 『:』が必要
※ 『for p in A :』でAのすべての要素に順にアクセスできる
※ 『for i in range(a, b) :』で i = a~b-1をループできる
※ for文はスコープを作らないのでfor文内で生成された変数をfor文の外でも参照できる

```
# ex6.py
import numpy as np

A = np.array([1,2,3,4,5,6,7,8,9,10])

sum = 0
for p in A :
    print( p )
    sum += p
    tmp = 5

print(sum)
print(tmp) #tmpも参照可能

sum = 0
N   = A.shape[0]
for i in range(N):
    sum += A[i]

print(sum)
```

Ex7.py　if文

- if 文は右のコードの通り定義できる
- for文と同様にブロックを定義する

右のコードを実行し挙動を確認してください

※ AかつB や AまたはBは 以下の通り

→ if(条件A and 条件B)：

→if(条件A or 条件B)：

※ if文はスコープを作らないのでif文内で生成された変数を外から参照できる

```
# ex7.py
import numpy as np

A = [1,2,4,2,1,1,3,4]
for p in A :
    if( p == 1 ) :
        print( "a" )
    elif( p == 2 ) :
        print( "b" )
    else :
        print ("c")
```

Ex8.py　関数

- 関数は右のコードの通り定義できる

右のコードを実行してください.

右のコードの中身を色々変化させ，変数のスコープを確認してください

※複数の引数を受け取れる

※引数は，値渡し（と思ってよい）

ある引数aがあるとき，aに代入をしない場合はaへの参照が保持される．関数内でaへの代入が起こると，その瞬間に新たに変数aが生成される．そのため，関数外部からみると引数変数の変化は起きないため，値渡しのように見える．

※複数の値を返せる

※関数はスコープを作る：関数内部で定義した変数は外に漏れない

※関数はスコープの外の変数を参照できる，（ただし関数内部で代入をすると，その変数は関数のローカル変数に）

```
# ex8.py
import numpy as np

def func(a,b) :
    print("a is ", a)
    print("b is ", b)
    wa = a+b
    sa = a-b
    print("c is ", c) #外のcを参照できる
    return wa, sa

a = 1
b = 2
c = 5
sum, sub = func(1,2)

print( a, b, sum, sub)
```

Ex9.py　クラス

- 今回は利用しないので割愛

PythonとOpenCVを利用した画像処理

- OpenCVとは
  - Open sourceの画像処理ライブラリ群
  - 多様な画像処理ツールを提供する
  - BSDライセンス：『「無保証」であることの明記と著作権およびライセンス条文自身の表示を再頒布の条件とするライセンス規定』(https://ja.wikipedia.org/wiki/BSDライセンス より)
- C++, Python, java, unityなどから利用可能

以降のコードは学内環境にて動作することを確認しています
もし途中で落ちる場合は画像データの読み込みに失敗している場合があります
ファイル名や配置するフォルダを確認してください

Ex10.py 画像の入出力

適当な画像を準備し，名前を「img.png」としてコードと同一フォルダに配置してください

右のコードを実行し画像が表示されることを確認してください

draw部分を変更し挙動を確認してください

※非常に手軽に画像読み込み、画像への書き込み，画像の表示，画像の書き出しが行なえます

※画像は np.array 形式で表現されます

cv2.line　　(画像, 点1, 点2, 色, 太さ)

cv2.rectangle(画像, 点1, 点2, 色, 太さ)

cv2.circle　　(画像, 中心, 半径, 色, 太さ)

※その他の図形描画関数は以下を参照
http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

```
# ex10.py
import numpy as np
import cv2

#load image
img = cv2.imread("img.png")
print( img.shape, type(img), img.dtype )

#draw rect and dots
cv2.line      (img,(100,100),(300,200), (0,255,255),2)
cv2.circle    (img,(100,100), 50,       (255,255,0),1)
cv2.rectangle(img,(100,100),(200,200), (255,0,255),1)

#display img
cv2.imshow("show image", img)
cv2.waitKey()

#save image
cv2.imwrite("img_save.png", img);
```

Ex11.py 画素へのアクセス

右のコードの一部を編集し画像をグレースケール化してください

※途中計算時のオーバフローを避けるため，画像 (img, img_gray)は一度float型に変換されています

```
# ex11.py
import numpy as np
import cv2

#load image
img = cv2.imread("img.png")
img = img.astype(np.float)
H   = img.shape[0]
W   = img.shape[1]
img_gry = np.zeros( (H,W) )

for y in range(H) :
    for x in range(W) :
        img_gry[y,x] = 128 #ここを編集

#display img
cv2.imshow("color image", np.uint8( img) )
cv2.imshow("gray  image",
np.uint8( img_gry) )
cv2.waitKey()
```

Ex12.py 線形フィルタ

右のコードの一部を編集し，縦横ソーベルフィルタを完成させてください

※途中計算時のオーバフローを避けるため，画像 (img, img_gray)は一度float型に変換されています

ヒント: マイナスの画素値は正値にして青chへ，正の画素値は赤chに入れてください

```
# ex12.py
import numpy as np
import cv2

#load image as grayscale and float
img = cv2.imread("img.png",0).astype(np.float)
img_x = np.zeros_like( img )
img_y = np.zeros_like( img )

for y in range(1, img.shape[0]-1) :
    for x in range(1, img.shape[1]-1) :
        dx = 128 #ここを編集
        dy = 128 #ここを編集

        if( dx < 0   ) : dx *= -1
        if( dx > 255 ) : dx  = 255
        if( dy < 0   ) : dy *= -1
        if( dy > 255 ) : dy  = 255
        img_x[y,x] = dx
        img_y[y,x] = dy

cv2.imshow( "a", img  .astype(np.uint8) )
cv2.imshow( "b", img_x.astype(np.uint8) )
cv2.imshow( "c", img_y.astype(np.uint8) )
cv2.waitKey()
```

Ex13.py 平均画像

右のコードの一部を編集し，３個の画像の平均画像を作成してください.

入力する画像は，同じサイズのものを適当に準備してください

※途中計算時のオーバフローを避けるため，画像 (img, img_gray)は一度float型に変換されています

```
# ex12.py
import numpy as np
import cv2

#load image and comvert to float
img1 = cv2.imread("img1.png").astype(np.float)
img2 = cv2.imread("img2.png").astype(np.float)
img3 = cv2.imread("img3.png").astype(np.float)

#以下を編集し平均画像を生成せよ
img  = np.zeros_like(img1);

#display img
cv2.imshow("img1", np.uint8( img1) )
cv2.imshow("img2", np.uint8( img2) )
cv2.imshow("img3", np.uint8( img3) )
cv2.imshow("img ", np.uint8( img) )
cv2.waitKey()
```

Ex14.py 顔認識

- 右はOpenCVが提供する顔認識関数を利用して，顔認識を行うコードです．

このコードを実行し挙動を確認してください

パラメータを変更して挙動の変化を確認してください

- 多少確証が低くとも顔らしい部分をすべてマークするよう，コードを書き換えてください
- share folderにおいてあるhaarcascade_frontalface_default.xmlファイルをローカルにコピーしてください。

```
# ex14.py
import numpy as np
import cv2

#load grayscale image
image_gray = cv2.imread("IMG_5349.jpg",0)

face_cascade =
cv2.CascadeClassifier( './haarcascade_frontalface
                        _default.xml')
facerect = face_cascade.detectMultiScale(
             image_gray, scaleFactor=1.1,
             minNeighbors=1, minSize=(100, 100),
             maxSize=(300,300))

for rect in facerect:
    print(rect)
    cv2.rectangle(image_gray,
tuple(rect[0:2]),tuple(rect[0:2]+rect[2:4]),
(255,255,255), thickness=4)

cv2.imwrite("res.png", image_gray)
cv2.imshow("test", image_gray)
cv2.waitKey()
```

まとめ

- 画像処理プログラミングの雰囲気を味合うために，Python & OpenCV環境で，初歩的なコードを書いてみた．

- このPython & OpenCV環境は，比較的手軽にプロトタイピングが行えるので，興味がある学生は是非学修を進めてほしい

→ 後期の高度情報演習では，比較的大変な課題を用意してお待ちしています