コンピュータビジョン プログラミング課題

担当：井尻敬

Update 2026/06/23

準備：

●Step0 : Pythonをインストールし、以下のコマンドを利用して opencvとnumpyをインストールしてください．仮想環境なども設定したい方は設定してください

```
> pip install opencv-python numpy
```

●Step1 : 講義ページより『cv_template.zip』を取得し解凍してください（フォルダをダブルクリック > 全て展開 を選択）
●Step2 : コマンドプロンプトを開き、解凍してできたcv_template フォルダをカレントディレクトリにしてください（cv_templateフォルダを開いて アドレスバーに「cmd」と打ち込むなど。）
●Step3 : 下記のコマンドを実行し、テンプレートを初期化してください

```
>python cv_starter.py
```

●Step6: 初期化キーの提出
実行に成功すると、以下の通り出力されます。

```
>python cv_starter.py
initialize successXXXXXXXXXXX.XXXXXXXXX
```

この出力された一行『initialize successXXXXXXXXXXX.XXXXXXXXX』をscombzの『テスト：課題準備』より提出してください。「XXXXXXXXXXX.XXXXXXXXX」には、数値が入ります。また、テンプレートの1行目にハッシュ値が入ります。この数値とハッシュ値を不正行為防止に利用するため、ここで初期化したテンプレートを必ず利用して解答して下さい（ファイル紛失などの際はご連絡ください）。また、テンプレート内の.pyファイルのヘッダは消さないでください。初期化キーの提出がない場合や初期化されたものと整合性のないファイルが提出された場合は、採点対象としないか、大幅な減点を行います。

提出時の注意 ：

- **仕様を守ってください。**採点自動化のため『フォルダ名・ファイル名が間違っているもの・入出力の仕様を満たさないもの』は評価できず0点扱いとなることがあります。 各課題について、コマンドライン引数の詳細な仕様など、入出力の仕様が指定されます。正しく従ってください。

- **動作確認をしてください。**各課題に入出力例が用意されています。必ず提出前に、自身で作成したプログラムを実行し正しく動作することを確認してください。（採点時には配布したものとは別のデータを使います）
- **極端に動作が遅いコードはNG。**本課題では計算速度を重視しませんが、極端に長い時間のかかるもの（20秒以上）は、自動採点の都合上0点とします。

その他の注意：

本課題および解答をweb上に公開しないでください。この課題の解答となるコードを、この課題の解答と分かる形でWeb上に公開する事は避けてください。また、ナリッジコミュニティサイト（知恵袋やteratailなど）にて、問題文をそのまま掲載し、解答を得る事は行なわないでください。

ソースコードのコピー禁止。学生同士が教え合うことは問題ありませんが、解答となるソースコードは必ず自分で作成して提出してください。解答となるソースコードの他者との共有や、ソースコードのコピーが発見された場合には、コピー元・コピー先の両名ともカンニングと同様の処置をとります。※Pythonはコードがどうしても似てしまうことは理解しています。していないカンニングの疑いをかけられないかと不安になったり、あえて”へんな書き方”をする必要はないです。

（将来AIを使って作業を加速するために）生成AIは使わないでください。この課題は、自然言語で書かれた課題を理解しプログラムに変換する訓練を行うことで、知識と経験の蓄積を目指すものです。そのため、ChatGPT等の生成系AIの利用は禁止します。

　このような比較的単純な課題を解く経験を蓄積することで、研究や仕事における実問題を解くときに『こういう課題ならWeb上にたくさんコードがあるのでAIもいい結果を出してくれそう』とか『このプロンプトを入れればこういうコードを出してくれそう』のような予測ができたりするなど、AIの支援を受けながら作業を加速する能力が身に付くと思います。

目次

※ 雛形に多くのヒントがあります。まずは雛形にあるコードを読んでください。

※ 発展課題は通常の課題が簡単すぎる方のためのものです。

※ 発展課題以外にも多少難易度の高い課題もあるので、全問解かなくてはいけないと気負わずに面白そうなものに取り組んでください（全課題を簡単に解いてしまう方が毎年数パーセントいるのも事実ではあるのですが。。）

○ Day 1 ○

締め切り：

scombz参照

提出方法:

1. 初期化した cv_template フォルダ内の day[1-5] というフォルダに解答を記述してください
   ※1日目なら day1というフォルダです
2. 作業した day[1-5]というフォルダをzip圧縮してください 「右クリック > 送る > 圧縮（zip）」
3. 出来上がった zipファイルをscombzより提出してください

※ 提出ファイル名は day[1-5].zip となるはずです。

※ 発展課題は、ファイル提出だけでなく教員（またはTA）に確認を受けてください。

※ 必ず初期化を行ったテンプレートを利用してください

テンプレート（雛形）：

講義ページ（https://takashiijiri.com/classes/cv/）に雛形があるので、必ず利用してください。テンプレートのファイル名や書くコードのヘッダは変更しないでください（課題XXのファイル名はexerXX.pyのまま）。初期化は最初に１回だけ実施してください。

01 Sum of Squared Difference

サイズの同じ2枚の画像を読み込み、グレースケール画像に変換後、2枚の画像間のSSD値を出力せよ。

- 入力ファイル名(img1.png、img2.png)はコマンドライン引数より取得せよ
  ※ 下は各課題の実行コマンドを示します

```
> python exer01.py img1.png img2.png
```

- 入力画像は、比較前にグレースケール画像に変換すること
- 計算したSSD値は標準出力に出力すること
- 標準出力には計算結果以外を出力しないこと

※ すべての課題にひな形 exerXX.pyが用意してあるので参考にしてください。

入出力例

```
> python exer01.py data/img1.png data/img2.png
2155324.0

> python exer01.py data/img1.png data/img3.png
12161624.0
```

day1/dataフォルダ内の img1.png と img2.png のSSDと
day1/dataフォルダ内の img1.png と img3.png のSSDは、上のとおりです。

※ 各課題の入出力例を提供します。画像ファイル等は雛形に同梱してあります。
※ 自身で作成したプログラムの出力と比較することでデバッグに利用してください。
※ 小さな数値の誤差は許容します。

02 PSNR（Peak signal-to-noise ratio）

サイズの同じ2枚の画像を読み込み、グレースケール化した後、2枚の画像間のMean Squared Error（MSE）とPeak signal-to-noise ratio（PSNR）を出力せよ。

- ふたつの入力ファイル名(img1.png、img2.png)はコマンドライン引数より取得せよ

```
> python exer02.py img1.png img2.png
```

- 入力画像は、比較前にグレースケール画像に変換すること
- MSE値を標準出力の1行目に、PSNR値を標準出力の2行目に出力すること
- 標準出力には，計算結果以外を出力しないこと
- PSNRとは、信号とノイズの比を表す数値のことであり、下記の通り計算できる。I1とI2は入力画像、Mは画像の取りうる最大値（ここでは255）、HとWは画像の高さと幅である。

$$\mathrm{PSNR}(I_1, I_2) = 10\log_{10}\frac{\mathrm{M}^2}{MSE(I_1, I_2)}$$

$$\mathrm{MSE}(I_1, I_2) = \frac{1}{\mathrm{WH}}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}\left(I_1(y,x) - I_2(y,x)\right)^2$$

ここでは、sklearn.metrics.mean_squared_error、cv2.PSNR()、skimage.metrics.peak_signal_noise_ratio()は利用せず、MSEやPSNRは自作すること。

入出力例

```
> python exer02.py data/img1.png data/img2.png
58.56858695652174
30.4541561463384

> python exer02.py data/img1.png data/img3.png
330.4789130434783
22.93936607306533
```

day1/dataフォルダ内のimg1.pngとimg2.png のMSEとPSNRや、
day1/dataフォルダ内のimg1.pngとimg3.png のMSEとPSNRは、上記の通りです。

※ PSNRは、例えば、『元画像』と『劣化画像』を入力として、劣化画像がどの程度元画像に近いかを評価するために利用されます。img2はimg1を50%に縮小してから元のサイズに拡大したもので、img3は20%に縮小してから拡大したものです。

03 Template Matching 1

ターゲット画像とテンプレート画像を読み込みTemplate Matchingを計算せよ。

- 入力ファイル名(target.png、template.png)と、出力ファイル名(output.png)はコマンドライン引数より取得せよ

```
> python exer03.py target.png template.png output.png
```

- ターゲット・テンプレート画像はグレースケール画像に変換してから計算すること
- 計算結果は各画素にSSD値を格納した画像として出力すること
  - 出力画像の画素(*i, j*)には、ターゲット画像の窓領域 [i:i+h, j:j+w]とテンプレート画像とのSSD値を記録すること
  - テンプレート画像を重ねられる領域を考慮し、ターゲット画像が HxW，テンプレート画像が hxwなら，出力画像サイズは(H-h+1) x (W-w+1) とせよ
- 出力画像は値域[0,255]の範囲へ正規化せよ（SSDの最大値で割り，255を掛ける）

※ ここではOpenCVの関数（matchTemplate()など）は利用せず，独自に実装すること

※ np.sumなどは使ってください．

入出力例

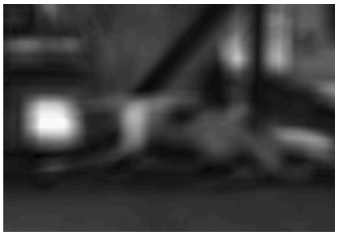
```
> python exer03.py data/target.png data/template1.png output_tm1_1.png
> python exer03.py data/target.png data/template2.png output_tm1_2.png
```

./data/target.png

./data/template1.png

./data/template2.png

./data/output_tm1_1.png



./data/output_tm1_2.png

day1/dataフォルダに入出力例となる画像があります。

target.pngとtemplate1.pngを利用してtemplate matchingをした結果→output_tm1_1.png

target.pngとtemplate2.pngを利用してtemplate matchingをした結果→output_tm1_2.png

04 Template Matching 2

ターゲット画像とテンプレート画像を読み込み、Template Matchingによりテンプレートと最も似た領域『**3か所**』を発見しその部分に矩形を描画せよ。

- ふたつの入力ファイル名(target.png template.png)と、出力ファイル名(output.png)は、下記の通り、コマンドライン引数より取得すること

```
> python exer04.py target.png template.png output.png
```

- ターゲット・テンプレート画像はグレースケール画像に変換してから計算すること
- 出力はカラー画像とし、ターゲット画像中のテンプレート画像に最も似た3個の領域にテンプレートと同じサイズの四角形を描画し出力すること
- SSD値が一番小さい位置，2番目に小さい位置，3番目に小さい領域を見つけること
- １番目の領域（四角形）と重ならないように２番目の領域を探索すること
- １，２番目に発見した領域と重ならないように、3番目の領域を探索すること
- 矩形描画には『cv2.rectangle (img, (x1,y1), (x2,y2),(b,g,r), line_width)』 を利用し、線幅2，線の色を青(255,0,0)とすること

※ OpenCVのmatchTemplate() と minMaxLoc() は利用せず独自に実装すること

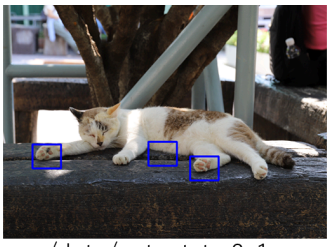
入出力例

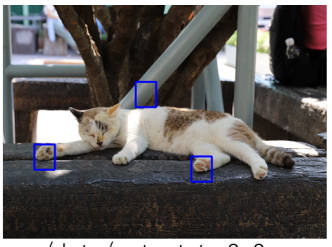
```
> python exer04.py data/target.png data/template1.png output_tm2_1.png
> python exer04.py data/target.png data/template2.png output_tm2_2.png
```


./data/target.png


./data/template1.png


./data/template2.png


./data/output_tm2_1.png


./data/output_tm2_2.png

day1/dataフォルダに入出力例となる画像があります。

target.pngからtemplate1.pngを探索した例→ output_tm2_1.png

target.pngからtemplate2.pngを探索した例→ output_tm2_2.png

05 劣化画像の復元 (発展)

ランダムに選択された30%の画素が黒く塗りつぶされた劣化画像を入力とし、なるべく元画像に近い画像を復元せよ。

- 下記の通り、入力ファイル名(input.png)、出力ファイル名(output.png)はコマンドライン引数より取得すること

```
> python exer05.py input.png output.png
```

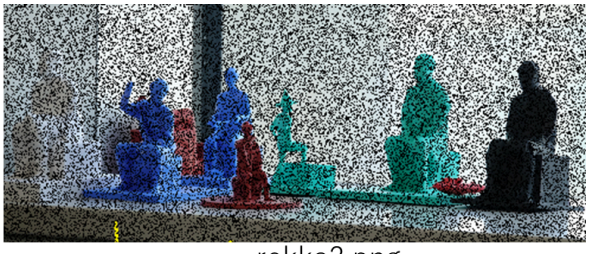
- 提出プログラムにより復元された画像と元画像とのPSNRが閾値T = 31.0 以上のものを正解とする

/day1/data フォルダ内にて、下記の元画像と劣化画像を配布します。劣化画像のみを入力として、元画像に近い画像を復元してください。 解法はいろいろあります。自分で考えて画像を復元してみてください。
※採点は違う画像で行います。
※PSNRの計算にはカラー画像を利用します。


orig1.png


rekka1.png


orig2.png


rekka2.png

※ OpenCVの修復用関数（`cv2.inpaint` など）や、その他画像修復用の外部関数は使用せず、NumPy等を用いてアルゴリズムを独自に実装すること

※余裕のある方向けの課題です
※この課題は、講義中にTAまたは教員の確認を受けてください。
※ コーディングの内容に関してごく簡単な口頭試問を行います。

○ Day 2 ○

締め切り：
scombz参照

提出方法:
1. 初期化した cv_template フォルダ内の day[1-5] というフォルダに解答を記述してください
   ※2日目なら day2というフォルダです
2. 作業した day[1-5]というフォルダをzip圧縮してください 「右クリック > 送る > 圧縮（zip）」
3. 出来上がった zipファイルをscombzより提出してください
※ 提出ファイル名は day[1-5].zip となるはずです。
※ 発展課題は、ファイル提出だけでなく教員（またはTA）に確認を受けてください。
※ 必ず初期化を行ったテンプレートを利用してください

テンプレート（雛形）：
講義ページ（https://takashiijiri.com/classes/cv/）に雛形があるので、必ず利用してください。テンプレートのファイル名や書くコードのヘッダは変更しないでください（課題XXのファイル名はexerXX.pyのまま）。初期化は最初に１回だけ実施してください。

06 Harrisのコーナー検出1 準備

画像と画素位置(x,y)を読み込み、その画素を中心とするサイズ5x5の窓領域における勾配を計算し出力せよ。

- 下記の通り、入力ファイル名(img_in.png)、画素位置(x_in y_in)、出力ファイル名(output.txt)はコマンドライン引数より取得すること

```
python exer06.py img_in.png x_in y_in output.txt
```

- 入力画像をグレースケール化してから計算すること
- 各画素における縦横方向微分には、右図のSobel filter（※ 重みを4で割ったもの）を利用せよ
- 計算した勾配をラスタスキャン順（右図）にテキストファイルへ出力せよ
- 出力ファイルでは、1行にひとつの勾配ベクトル『x成分 y成分』を書き，x成分とy成分の間には半角スペースを配置せよ

※ 出力の詳細は雛形および解答例を確認すること

-1/4	-2/4	-1/4
0	0	0
1/4	2/4	1/4

-1/4	0	1/4
-2/4	0	2/4
-1/4	0	1/4

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

※「12」を注目画素として、その周囲領域0~24における勾配ベクトルを出力する

入出力例

```
> python exer06.py data/img_cat.png 216 75 output_hr1_1.txt
> python exer06.py data/img_cat.png  85 35 output_hr1_2.txt
```

/day2/dataフォルダに入出力例があります。
./data/img_cat.png について、
画素(216, 75) 付近の勾配が output_hr1_1.txt に、画素(85, 35) 付近の勾配が output_hr1_2.txt にあります。(216, 75)はコーナー付近の画素、(85, 35)はエッジ付近の画素です。

./data/img_cat.png

07 Harrisのコーナー検出 2 行列作成

画像と画素位置(x,y)を読み込み、その画素位置におけるHarris行列を計算し出力せよ。

- 下記の通り、入力ファイル名(img_in.png)、画素位置(x_in y_in)、出力ファイル名(output.txt)はコマンドライン引数より取得すること

```
python exer07.py img_in.png x_in y_in output.txt
```

- 入力画像はグレースケール化してから計算すること
- 計算には、右上のsobel filter（重み*1/4）と右下のGaussian重みを利用せよ
- 計算したHarris 行列 (2x2行列)は，テキストファイルへ2行で出力すること（出力例参照）

-1/4	-2/4	-1/4
0	0	0
1/4	2/4	1/4

-1/4	0	1/4
-2/4	0	2/4
-1/4	0	1/4

$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{6}{256}$	$\frac{24}{256}$	$\frac{36}{256}$	$\frac{24}{256}$	$\frac{6}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$

Gaussian 重み

※ OpenCVの関数（cv2.cornerHarrisなど）は利用せず自作すること

※入力画像のふち付近では5x5の窓領域がはみ出し、Harris行列が計算できない．このようなふち画素は指定されないものと仮定してよい

入出力例

```
> python exer07.py data/img_cat.png 216 75 output_hr2_1.txt
> python exer07.py data/img_cat.png  85 35 output_hr2_2.txt
```

day2/dataフォルダに入出力例があります。

./data/img_cat.png について、

画素(216, 75)のHarris行列が output_hr2_1.txt に、

画素(85, 35)のHarris行列が output_hr2_2.txt にあります。

(216, 75)はコーナー付近の画素、(85, 35)はエッジ付近の画素です。

./data/img_cat.png

08 Harrisのコーナー検出 3

Harrisのコーナー検出法により入力画像からコーナーを検出し、コーナーに円を描画した画像を出力せよ。

- 下記の通り、入力ファイル名（fname_in.png）と出力ファイル名（fname_out.png）はコマンドライン引数より取得すること

```
python exer08.py fname_in.png fname_out.png
```

- Harris行列計算の仕様は前問の通り
- 評価式 R は，Harris行列の固有値 λ_1, λ_2を用いて以下の通り計算せよ

$$R = \lambda_1\lambda_2 - 0.15 * (\lambda_1 + \lambda_2)^2$$

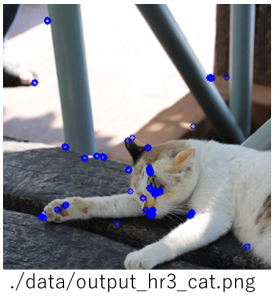
- Opencvの関数（cv2.cornerHarris）は利用せず，行列計算・評価式Rの計算部分は自作すること
- R≧260,000の画素をコーナーとして検出し，検出画素を中心とした円（半径3・色(255,0, 0)・線幅1）をcv2.circle() 関数を利用して描画した画像を出力すること

入出力例

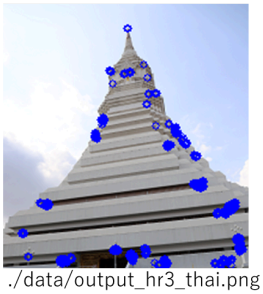
```
> python exer08.py data/img_cat.png  output_hr3_cat.png
> python exer08.py data/img_thai.png output_hr3_thai.png
```

./dataフォルダ内に入出力例となる画像があります。以下の通りコーナーに点を描ければOKです。


./data/img_cat.png


./data/output_hr3_cat.png


./data/img_thai.png


./data/output_hr3_thai.png

09 Canny Filter

画像を読み込みCanny Filterによりエッジ画像を生成し出力せよ。

- 下記の通り、入力・出力ファイル名はコマンドライン引数により取得すること

```
python exer09.py img_in.png img_out.png
```

- OpenCVの関数「cv2.Canny」を利用すること
- 二つの閾値 TmaxとTminは，それぞれ，165と85とすること
- 勾配の計算には 3x3 Sobelフィルタを利用すること：デフォルトのまま
- 勾配強度は L2gradient (L2ノルム)を利用すること：デフォルトではない

※ 関数の使い方を検索して使う練習をするのがこの問題の出題意図です

入出力例

```
> python exer09.py data/img_cat.png  output_canny_cat.png
> python exer09.py data/img_thai.png output_canny_thai.png
```

./dataフォルダ内に入出力例となる画像があります。以下の通り細いエッジを含む画像を生成できればOKです。


./data/img_cat.png


./data/output_canny_cat.png


./data/img_thai.png


./data/output_canny_thai.png

10 Seam Carving (発展)

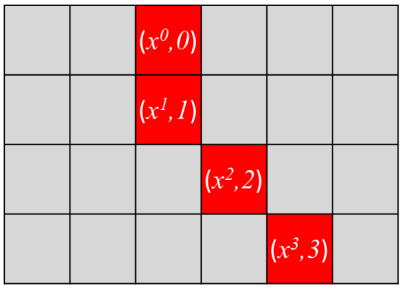
Seam Carvingを行うプログラムを実装せよ

- 下記の通り、入力画像名をコマンドライン引数として取得すること

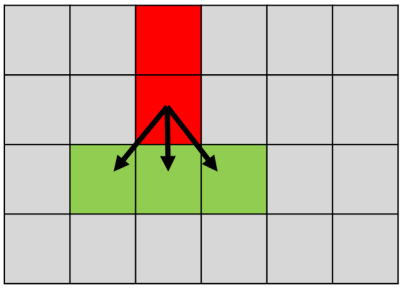
```
python exer10.py img.png
```

- Seam Carvingアルゴリズムについては『 "Seam carving for content-aware image resizing | ACM SIGGRAPH 2007". Siggraph 2007』 を参照
- Seam Carving関数を除いたコードを雛形として配布するので利用すること
- aキーを押すと，Seam Carvingを行い画像を横方向に1画素分縮小する
- bキーを押すと，現在の画像が[out.png]という名前で保存される
- 削除する画素の重要度e(x,y)は，論文中の式(1) $\left|\frac{\partial I(x,y)}{\partial x}\right| + \left|\frac{\partial I(x,y)}{\partial y}\right|$ を利用すること（Iは輝度値画像）
- skimage.transform.seam_carveなどの外部関数は利用せず，seam探索については自作すること

Seam carvingアルゴリズムの解説



- 例としてサイズ 6 x 4 の小さな画像を、seam carvingにより横方向に1画素分小さくする問題を考える
- 画素(x,y)には，重要度 $e(x,y)$ が定義されている
- 検索したいシームが満たす条件は。。。
  - 上端の画素と下端の画素をつなぐ
  - ある画素からひと画素分下に移動するとき，左隣・真下・右隣の3通りに移動できる
  - 上記2条件を満たすもののうち、重要度の総和が最小となる



- このようなシームは動的計画法により高速に計算可能
  - Step1 : ラスタ順に各画素をたどり、『上端から着目画素までたどる際の重要度の総和』と『着目画素の上段の画素の位置』を記録する
  - Step2 : 下端の画素のうち最も重要度の総和が小さいものを発見し、『着目画素の上段の画素の位置』の情報を利用して上方向にたどりシームとする

※ より詳しい資料を講義資料ページに置いておくので参考にしてください。

※ かなり余裕のある方向けの発展課題です

※ 解けたら、講義中にTAか教員に確認をうけてください

○ Day 3 ○

締め切り：

scombz参照

提出方法:

1. 初期化した cv_template フォルダ内の day[1-5] というフォルダに解答を記述してください
   ※1日目なら day1というフォルダです
2. 作業した day[1-5]というフォルダをzip圧縮してください 「右クリック > 送る > 圧縮（zip）」
3. 出来上がった zipファイルをscombzより提出してください

※ 提出ファイル名は day[1-5].zip となるはずです。

※ 発展課題は、ファイル提出だけでなく教員（またはTA）に確認を受けてください。

※ 必ず初期化を行ったテンプレートを利用してください

テンプレート（雛形）：

講義ページ（https://takashiijiri.com/classes/cv/）に雛形があるので、必ず利用してください。テンプレートのファイル名や書くコードのヘッダは変更しないでください（課題XXのファイル名はexerXX.pyのまま）。初期化は最初に１回だけ実施してください。

11 特徴ベクトル1

画像と窓情報「左上の画素 (x, y)、幅w、高さh」を受け取り、画像をグレースケール化した後、窓内のヒストグラムを計算しテキストファイルに出力せよ。

- 下記の通り、入力・出力ファイル名はコマンドライン引数により取得すること

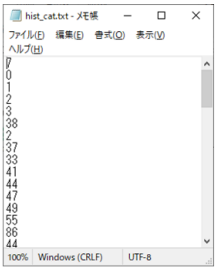
```
python exer11.py img_in.png x y w h out.txt
```

  - img_in.png : 入力画像
  - x y : 窓領域の左上画素位置を示す
  - w h : 窓領域の幅と高さを示す
  - out.txt : 出力ファイル
- ヒストグラムについて
  - 『画素値 = 画素値 // 4 』という計算により画素値の階調数を [0, 63] に変化させ、64階調のヒストグラムを計算すること
  - ヒストグラムの出力内容は、入出力例を参考にすること
- 必ず np.histogramを利用すること。for文を利用した自作はしてはならない（遅くて自動採点できないので）
- np.histogramには binsとrangeを正しく指定してください

入出力例

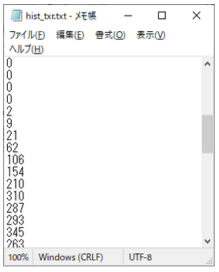
```
> python exer11.py data/img_cat.png 250 100 40 40 hist_cat.txt
> python exer11.py data/img_txr.png  10  80 50 50 hist_txr.txt
```

data/img_cat.png



data/hist_cat.txt

data/img_txr.png



data/hist_txr.txt

※ 次の課題ではこれを特徴ベクトルとして利用します。

12 特徴ベクトル2

ターゲット画像とテンプレート画像を読み込み、Template Matchingによりテンプレートと最も似た領域を発見しその部分に矩形を描画せよ。

- ふたつの入力ファイル名(target.png template.png)と、出力ファイル名(output.png)は、下記の通り、コマンドライン引数より取得すること

```
> python exer12.py target.png template.png output.png
```

- ターゲット・テンプレート画像はグレースケール画像に変換してから計算すること
- 出力はカラー画像とし、ターゲット画像中のテンプレート画像に最も似た領域にテンプレートと同じサイズの四角形を描画し出力すること
- 矩形描画には『cv2.rectangle (img, (x1,y1), (x2,y2),(r,g,b), line_width)』 を利用し、線幅2，線の色(255,0,0)とすること
- 画像の類似度には、ヒストグラムの差を利用すること
  - ターゲット画像の矩形領域（テンプレート画像と同じサイズ）のヒストグラムと、テンプレート画像のヒストグラムの自乗誤差を相違度として利用すること
  - ヒストグラム作成時『画素値 = 画素値 // 4 』という計算により画素値の階調数を[0, 63] に変化させ、64階調のヒストグラムを計算すること
  - 必ず np.histogramを利用すること。for文を利用した自作はしてはならない

入出力例

```
> python exer12.py data/img_cat.png data/temp_cat.png output_cat.png
> python exer12.py data/img_txr.png data/temp_txr.png output_txr.png
```

dataフォルダに入出力例となる画像があります。
img_cat.png から temp_cat.pngを探索した例→ output_cat.png
img_txr.png から temp_txr.png を探索した例→ output_txr.png


data/img_cat.png


temp_cat.png


data/output_cat.png


data/img_txr.png


temp_txr.png


data/output_txr.png

※ テンプレートが回転していても正しい場所を特定できていることに注意してください

13 Hough変換 1

画像を読み込み以下の手順でHough変換画像を計算せよ。

1. 下記の通り、入力ファイル名（img_in.png）と出力ファイル名（img_out.png）をコマンドライン引数により取得

```
python exer13.py img_in.png img_out.png
```

2. 入力画像をグレースケール画像化し、勾配強度画像を計算する（右図のsobel filterにより勾配強度を計算した後、最大値で全体を除算し[0,1]に正規化する）
3. 勾配強度画像を閾値により二値化（値0.35以上を前景に）
4. 全ての前景画素(y, x) について、Hough変換画像へ投票する
   - θ の値域は0~359とする（1画素の幅が1度に対応）
   - ρ の値域は0~A-1とする（1画素の幅が1画素分に対応、Aは画像の対角方向の長さ）
   - 投票方法は講義中に解説したものを利用すること
   - pは int()で切り捨てること。また、theta = t / 360 * 2 * math.pi　を利用してradianを求めること。
5. Hough変換画像を、画像内の最大値で除算して255をかけることで[0,255]に正規化する

-1/4	-2/4	-1/4	-1/4	0	1/4
0	0	0	-2/4	0	2/4
1/4	2/4	1/4	-1/4	0	1/4

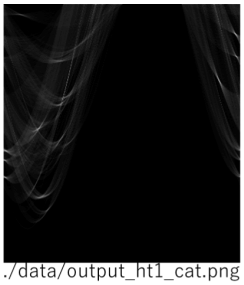
Sobel filter

入出力例

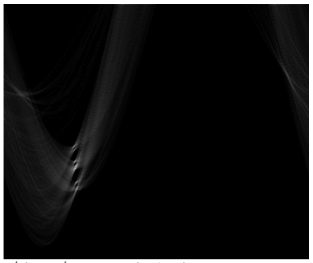
```
> python exer13.py data/ht_cat.png  output_ht1_cat.png
> python exer13.py data/ht_thai.png output_ht1_thai.png
```

dataフォルダ内に入出力例となる画像があります。以下の通りhough変換した画像を作成できればOKです。


./data/ht_cat.png


./data/output_ht1_cat.png


./data/ht_thai.png


./data/output_ht1_thai.png

14 Hough変換 2

入力画像を読み込み、前課題で作成したHough変換画像を利用して直線を検出し、検出した直線を入力画像に描画せよ。

- 下記の通り、入力ファイル名（img_in.png）と出力ファイル名（img_out.png）はコマンドライン引数より取得すること

```
python exer14.py img_in.png img_out.png
```

- 前の課題の手順でHough変換画像を作成し、Hough変換画像は正規化せず，投票数が80以上の(ρ, θ) の組に対する直線を描くこと
- 出力はカラー画像とし、直線はcv2.line()関数を利用して『色(B=255, G=0, R=0)，線幅1』のものを描くこと

※ cv2.HoughLines(), cv2.HoughLinesP()を利用せず、Hough変換画像の生成や直線の検出部分は自作すること

※ 正解データは、画像の左右の端の2点を結ぶ直線を描くことで作成しています

入出力例

```
> python exer14.py data/ht_cat.png  output_ht2_cat.png
> python exer14.py data/ht_thai.png output_ht2_thai.png
```

./dataフォルダ内に入出力例となる画像があります。以下の通り直線が複数引ければOKです。計算方法により多少の誤差がでるので、ある程度の誤差を許容する形で採点します。


./data/ht_cat.png


./data/output_ht2_cat.png


./data/ht_thai.png


./data/output_ht2_thai.png

15 トランプのスート識別（発展）

トランプのスート（ダイア・ハート・クラブ・スペード）を撮影した画像を読み込み、独自に設計した特徴量と閾値処理により、画像内のスートを識別せよ。下記の通り、入力ファイル名（img_in.png）はコマンドライン引数より取得すること

```
python exer15.py img_in.png
```

- 入力画像は、その中心にスートが来るように撮影されるものとする。
- 特徴ベクトルは独自に設計すること
- 識別のための閾値も独自に設計すること
- 識別結果の頭文字を標準出力に出力すること
    - ダイア → D、クラブ → C
    - ハート → H、スペード → S

識別対象のサンプル画像はdataフォルダにあります。

```
> python exer15.py data/suit1.png
H
> python exer15.py data/suit2.png
C
> python exer15.py data/suit3.png
S
> python exer15.py data/suit4.png
C
```

通常はサンプル画像を利用してパラメータをチューニングした後に未知画像の識別を行いますが、今回は、サンプル画像を利用してチューニングするだけでOKです。TAまたは教員の確認を受けてください。（特徴ベクトルの設計方針や閾値のチューニング方法を伺います）

○ Day 4 ○

締め切り：
scombz参照

提出方法:
1. 初期化した cv_template フォルダ内の day[1-5] というフォルダに解答を記述してください
   ※4日目なら day4というフォルダです
2. 作業した day[1-5]というフォルダをzip圧縮してください 「右クリック > 送る > 圧縮（zip）」
3. 出来上がった zipファイルをscombzより提出してください
※ 提出ファイル名は day[1-5].zip となるはずです。
※ 発展課題は、ファイル提出だけでなく教員（またはTA）に確認を受けてください。
※ 必ず初期化を行ったテンプレートを利用してください

テンプレート（雛形）：
講義ページ（https://takashiijiri.com/classes/cv/）に雛形があるので、必ず利用してください。テンプレートのファイル名や書くコードのヘッダは変更しないでください（課題XXのファイル名はexerXX.pyのまま）。初期化は最初に１回だけ実施してください。

16 Otsu Method

画像を読み込み、グレースケール画像に変換後、Otsu法により画像を二値化せよ。

- 下記の通り、入力ファイル名と出力ファイル名はコマンドライン引数より取得せよ

```
python exer16.py input.png output.png
```

- グレースケール画像の階調数は256 [0,255]とし、Otsu法適用のためのヒストグラムは，256階調 [0, 255]のものを利用せよ
- 出力は二値化画像とし，前景画素は255，背景画素は0とすること

入出力例

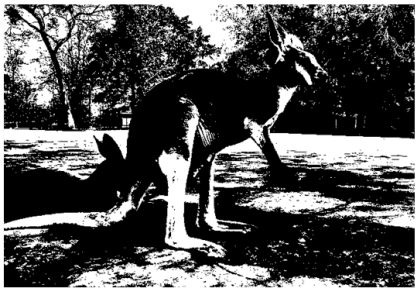
```
> python exer16.py data/kang.png output_otsu_kang.png
> python exer16.py data/cat.png  output_otsu_cat.png
> python exer16.py data/late.png output_otsu_late.png
```

3件の画像ファイルとそれらの二値化した結果がdataフォルダ内にあります。

./data/kang.png

./data/cat.png

./data/late.png



./data/output_otsu_kang.png

./data/output_otsu_cat.png

./data/output_otsu_late.png

17 Adaptive thresholding

画像を読み込み、グレースケール画像に変換後、Sauvola法により画像を二値化せよ。

- 下記の通り、入力ファイル名と出力ファイル名はコマンドライン引数より取得せよ

```
python exer17.py input.png output.png
```

- 画素 (x, y) の しきい値 T(x, y) は下記の通り計算すること
  - (x,y)を中心とする、11 x 11 の窓領域を作成する（上下左右5画素分）
  - 窓内の画素値の『平均 m(x,y)』と『標準偏差s(x, y)』を用いてしきい値を計算

$$T(x,y) = m(x,y)\left(1 + 0.2\left(\frac{s(x,y)}{255} - 1\right)\right)$$

- しきい値T以上の画素を255、それ以外の画素を0とした二値画像を出力すること
- 窓を作成できない画像の端 5画素分は、計算せず値を0とすること

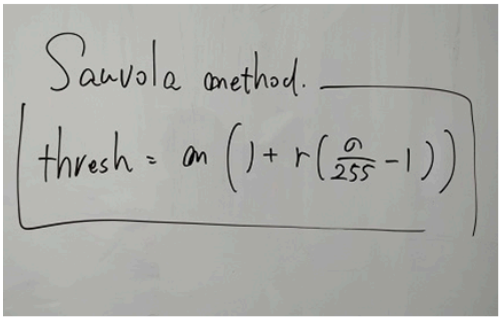
※ np.mean()や、np.var() or np.std()を 使って実装してみてください。

入出力例

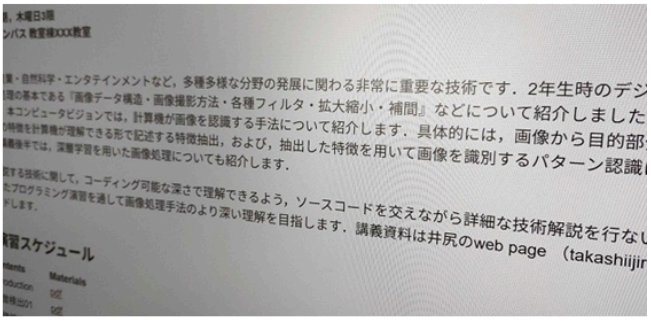
```
> python exer17.py data/moji1.jpg output_moji1.png
> python exer17.py data/moji2.jpg output_moji2.png
```

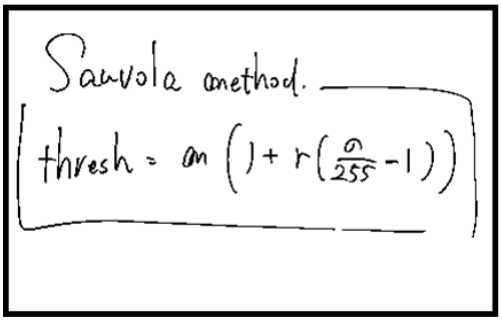
2件の画像ファイルとそれらの二値化した結果がdataフォルダ内にあります。

data/moji1.jpg

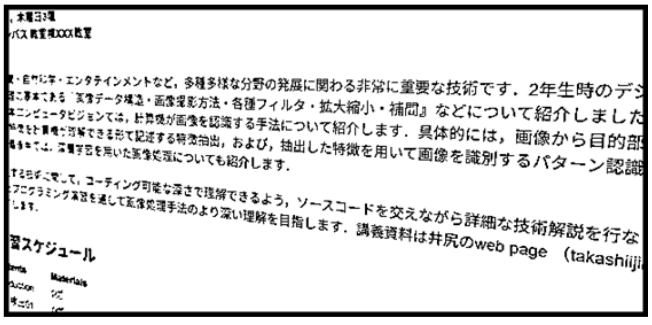


data/moji2.jpg





data/output_moji1.jpg



data/output_moji2.jpg

18 Region Growing

画像・シード画素位置・閾値を読み込み、画像をグレースケール化後，シードより領域成長を行い画像を二値化せよ

- 下記の通り、入力画像（img.png）、シード位置（seed_x seed_y）、閾値（t）、出力ファイル名（output.png）はコマンドライン引数より取得すること

```
python exer18.py img.png seed_x seed_y t output.png
```

- グレースケール画像の階調数は[0,255]とする（雛形参照）
- 出力は二値化画像とし，前景は白(255)，背景は黒(0)とすること
- 領域成長時について
  - ある画素について、その上下左右４個の画素を隣接しているとみなすこと
  - 成長中の領域に隣接する画素のうち，**閾値 *t* 以上** のものを加えること

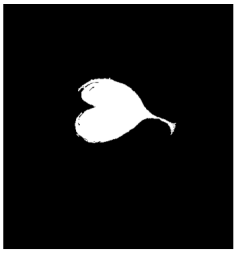
  ※ここで行う領域成長法は、講義中に説明した領域成長法（二値化）なので注意してください

入出力例

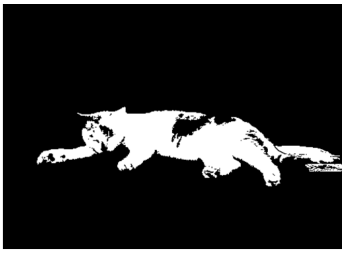
```
> python exer18.py data/late.png 180 180 140 output_rg_late.png
> python exer18.py data/cat.png  200 180 120 output_rg_cat.png
```

上記2件のコマンドにより、2件の画像ファイルに対してRegion Growingを適用した結果がdataフォルダ内にあります。


./data/late.png


./data/output_rg_late.png


./data/cat.png


./data/output_rg_cat.png

19 領域を数える

画像内の種の個数を数え標準出力に書き出すプログラムを作成せよ。

- 下記の通り、入力画像はコマンドライン引数より取得すること

```
python exer19.py img.png
```

- 入力画像について
  - サンプル画像(seeds1.jpg,seeds2.jpg)を配布する
  - 入力画像は、サンプル画像とほぼ同じ角度・照明条件で撮影される
  - 種どうしがなるべくバラバラになるよう撮影するが，種同士の多少の接触は残る可能性がある
- 画像内の種の数を標準出力に出力すること（標準出力には種の数以外は出力しない）
- 提出されたコードをテスト画像（非公開）に対して適用し，そのエラーが2個以内であれば正解とする（100個の種を含む画像に対して 98 ~ 102を出力すれば正解）

入出力例

```
python exer19.py data/seeds1.jpg
37
python exer19.py data/seeds2.jpg
122
```

dataフォルダ内にテスト用の種画像（seeds1.jpg と seeds2.jpg）があります。

※小松菜の種です。

20 独自のアプリ開発 (発展)

授業中に紹介した画像・動画処理手法である「YOLO」「SAM」「RAFT」「MediaPipe」から二つを選び、これらを活用した実用的なアプリケーション（ゲームも可）を設計し、実装せよ。

- 必ず、二つ以上の技術を組み合わせて利用すること（片方の出力をもう片方の入力に活用するなど。例) YOLOで検出した物体の軌跡をRAFTを用いて取得するなど）
- 必ず以下のどちらかを満たすこと
    - マウス・キーボードではなく、映像を使ってインタラクティブに操作するアプリケーション
    - 動画情報を取得し、その内容について意味のある解析・分析を行うアプリケーション

※ 他の課題が簡単すぎる方向けの課題です。
※ SAM / RAFTは、GPU環境がないと動かない可能性があります。自身の環境を確認してください。
※ TAまたは教員の確認を受けてください。

○ Day 5 ○

締め切り：

scombz参照

提出方法:

1. 初期化した cv_template フォルダ内の day[1-5] というフォルダに解答を記述してください
   ※5日目なら day5というフォルダです
2. 作業した day[1-5]というフォルダをzip圧縮してください 「右クリック > 送る > 圧縮（zip）」
3. 出来上がった zipファイルをscombzより提出してください

※ 提出ファイル名は day[1-5].zip となるはずです。

※ 発展課題は、ファイル提出だけでなく教員（またはTA）に確認を受けてください。

※ 必ず初期化を行ったテンプレートを利用してください

テンプレート（雛形）：

講義ページ（https://takashiijiri.com/classes/cv/）に雛形があるので、必ず利用してください。テンプレートのファイル名や書くコードのヘッダは変更しないでください（課題XXのファイル名はexerXX.pyのまま）。初期化は最初に１回だけ実施してください。

準備：MNISTデータの読み込み方法

MNISTデータ

- パターン認識の勉強によく利用される手書き数字画像のデータセット
- 画像サイズ 28x28（数字は画像の中心に配置される）
- データ数：トレーニング用60000文字、テスト用10000文字
- 形式：データは独自のバイナリ形式
- 本家のURL：http://yann.lecun.com/exdb/mnist/

MNISTデータの読み込み方法

1. http://yann.lecun.com/exdb/mnist/ からデータをダウンロード（アクセスできない場合、別の方法を授業中に指示します。）

```
train-images-idx3-ubyte.gz : 60000個のTraining data (画像)
train-labels-idx1-ubyte.gz : 60000個のTraining data (ラベル)
t10k-images-idx3-ubyte.gz  : 10000個のTest data (画像)
t10k-labels-idx1-ubyte.gz  : 10000個のTest data (ラベル )
```

2. 画像データの読み込み – バイナリ形式で読みこみ，行列の形に整形する

```
def open_mnist_image(fname) :
    f = gzip.open(fname, 'rb')
    data = np.frombuffer( f.read(), np.uint8, offset=16)
    f.close()
    return data.reshape((-1, 784)) # (n, 784)の行列に整形，nは自動で決定
```

3. ラベルデータの読み込み　– バイナリ形式ですべて読んで，1次元配列の形に整形する

```
def open_mnist_label(fname):
    f = gzip.open(fname, 'rb')
    data = np.frombuffer( f.read(), np.uint8, offset=8 )
    f.close()
    return data.flatten() # (n, )の1次元配列に整形，nは自動で決定
```

sklearnの準備

下記のコマンドを打ち込みsklearnをインストールしてください。

```
> python -m pip install scikit-learn
```

21 MNISTの読み込み

MNISTのトレーニングデータを読み n番目の画像とラベルを出力せよ。

- 下記の通り、整数値nはコマンドライン引数より取得すること

```
python exer21.py n
```

- ソースファイル（exer21.py）があるフォルダのひとつ上のフォルダに『mnist』という名前のフォルダを作成し，MNISTデータはそこから読むこと（パスはプログラム内にハードコードすること、雛形参照）

  path: ../mnist/train-images-idx3-ubyte.gz

  path: ../mnist/train-labels-idx1-ubyte.gz
- n番目の画像をpng画像として出力し，ファイル名は『n_[label値].png』とすること
- 例，n=20の画像のラベル値が4なら，ファイル名は『20_4.png』となる
- MNISTデータにアクセスできない場合ダウンロード方法を別途指示します。

入出力例

```
> python exer21.py 25
> python exer21.py 28
> python exer21.py 32
```

上記のように、25番目、28番目、32番目の訓練データを指定すると、下記のような画像が保存されます。ファイル名にも注意してください。
上記コマンドによる出力ファイル例が data フォルダにあります。

25_2.png

28_2.png

32_6.png

22 kNNによる文字認識

『MNISTのトレーニングデータ』と『ラベル値の不明な画像3枚』を読み込み、kNNによりラベルの値を推定せよ。

- 下記の通り、kNNのkの値（k）、3枚の入力画像名（img[1/2/3].png）はコマンドライン引数により取得すること

```
python exer22.py k img1.png img2.png img3.png
```

- MNISTの読み込みには前課題のものを利用すること（ソースファイルがあるフォルダのひとつ上のフォルダに『mnist』フォルダを作成し、データはそこから読む）
- 入力画像は，28x28のグレースケール画像（背景黒、文字が白）とする
- 推定結果となる3つの数値は，**標準出力に半角スペースで区切って出力すること**
- kNNは sklearnの KNeighborsClassifierを利用すること（使い方は各自webを検索，又は，雛形に例を載せておくので参照のこと）
- 全データを使うと処理が重いので、下記の通り訓練データを5000個に縮小すること

```
knn.fit( x_train[0:5000], t_train[0:5000])
#knnのところには皆さんが定義した変数名が入ります。
```

入出力例


./data/img1.png


./data/img2.png


./data/img3.png

data フォルダに、3枚のテスト画像（下記）が配置してあります。これらに対してmnistデータセットを利用してkNN（k=3）で推論をすると、『2 8 5』と正しく推論されます。

```
>python exer22.py 3 data/img1.png data/img2.png data/img3.png
2 8 5
```

23 主成分分析 1

3次元空間に分布する500個のデータ点群がある。このデータを読み込み、平均ベクトルと分散共分散行列を計算しテキストデータとして出力せよ。

- 入力データはコマンドライン引数により取得すること

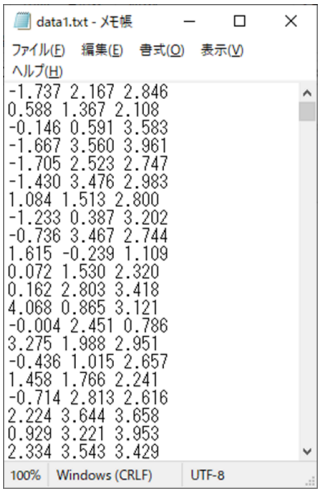
```
python exer23.py input_data.txt output.txt
```

    - input_data.txt : 入力ファイル名（500個の点群情報を含む入力データ）
    - output.txt : 出力ファイル名
- 入力ファイルの形式については、dataフォルダを参照のこと
- 平均ベクトル・分散共分散行列は、np.round() 関数を用いて、小数第3位を四捨五入して小数第2位までを出力すること（フォーマットについては出力例を参考のこと）
- 分散共分散行列の計算の際、分母は Nで割ること（不変分散（np.cov）ではなく、標本分散を利用してください）

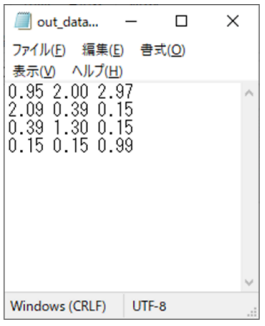
入出力例

data フォルダに、2個のテストデータがあります。

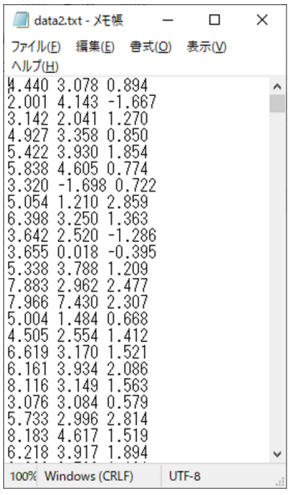
```
>python exer23.py data/data1.txt out_data1.txt
>python exer23.py data/data2.txt out_data2.txt
```



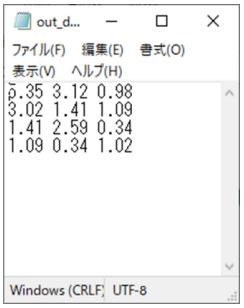
data/data1.png



data/out_data1.png



data/data2.png



data/out_data2.png

24 主成分分析 2

以下の手順にて、MNIST画像に主成分分析を実施した結果を画像として出力せよ

1) MNISTのトレーニングデータのうち『0番目から9999番目までのもの』かつ『ラベルが a または b のもの』のみを抽出する
2) 抽出した手書き文字画像を784次元ベクトルとみなして主成分分析を実施する
3) 第1主成分・第2主成分・第3主成分ベクトルを 28x28 の画像にして出力する

- a / b および 出力ファイル名はコマンドライン引数により取得すること

```
python exer24.py a b output1.png output2.png output3.png
```

  a) output1.png : 出力ファイル名（第1主成分の画像）
  b) output2.png : 出力ファイル名（第2主成分の画像）
  c) output3.png : 出力ファイル名（第3主成分の画像）

- MNISTの読み込みには前課題のものを利用すること（ソースファイルがあるフォルダのひとつ上のフォルダに『mnist』フォルダを作成し、データはそこから読む）
- 固有値固有ベクトルを求めるには『evals, evecs = np.linalg.eigh(mat) 』を使うこと（固有値が小さい順に並ぶので注意）
- 可視化する軸 v（784次元ベクトル）は正規化されているため、軸に 0.5を足し、255をかけてから画像に戻すこと
  - v = np.clip((v + 0.5) * 255, 0 , 255)
  - v = v.reshape(28,28)

dataフォルダに、(a,b) = (1,3)の結果と、(2,8)の結果があります。

※ このPCAの軸画像は何を表しているでしょうか？自分なりの解釈をしてみてください。

a=1　b=3

data/out13_1.png　data/out13_2.png　data/out13_3.png

a=2　b=8

data/out28_1.png　data/out28_2.png　data/out28_3.png

お疲れ様でした！