機械工学総合演習第二　流れの可視化計測

杵淵 郁也 * Mouterde Timothée † 李 敏赫 ‡ 渡村 友昭 §

2024 年 5 月 20 日

1　安全上の注意

本実験では高出力レーザを使用する．間違った使用により即失明する可能性がある．レーザ点灯中にはレーザ保護眼鏡を着用して作業を行い，実験中は教員& TA の指示に従うこと．なお，実験中にスマートフォン等を操作すると，危険の察知・判断を即座に行えない．自身の安全と班員の安全に十分に配慮すること．本実験では水を用いる．電子機器の水没や損傷を避けるため，実験室へ**紙媒体の実験ノート**を持参し，記録をとることを強く推奨する．

2　テキストエディタ

gedit を使用することを強く推奨する．

3　必要なソフトウェア

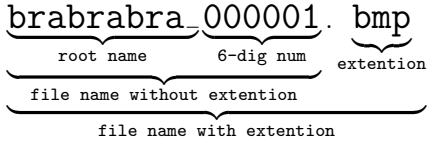
本実験では python とそのライブラリを用い，画像処理を行う．また，画像の閲覧は ImageJ を用いる．これらのソフトウェアがインストールされていない場合には，以下のコマンドを入力しインストールを行う．

How to execute 00

```
$␣pip␣list ↵
$␣pip␣install␣pillow ↵
$␣pip␣install␣matplotlib ↵
$␣which␣imagej ↵
$␣sudo␣apt-get␣install␣imagej ↵
$␣sudo␣apt-get␣install␣python3-tk ↵
```

4　ファイルに名に関するおさらい

コンピュータ上でファイルを扱う場合，正確な呼称を使う必要がある．今，barabrabra_000001.bmp というファイルについて，呼称を確認しよう．ファイルの拡張子（extention）は bmp である．拡張子有りのファイル名は barabrabra_000001.bmp である．拡張子無しのファイル名は barabrabra_000001 である．連続した 6 桁の番号は 000001 である．ファイルのルート名は barabrabra_である．アンダーバー_やドット．の取り扱いに注意されたい．



* E-MAIL: kine@fel.t.u-tokyo.ac.jp
† E-MAIL: mouterde@g.ecc.u-tokyo.ac.jp
‡ E-MAIL: mlee@mesl.t.u-tokyo.ac.jp
§ E-MAIL: watamura@g.ecc.u-tokyo.ac.jp

5　画像の基礎

非圧縮画像の構造は二次元の配列として表すことができ，白黒 8 bit 画像の輝度値 f は 0 から 255（256 階調）の値で濃淡が決められている．座標の原点は画像の左下にあり，表 1 の様な構造をしている．ここで，横軸を x は縦軸を y 方向と定義すると，輝度値 f は $f(x,y)$ と記述できる．ただし，ImageJ で画像を表示すると，原点が左上になるので，注意すること．また，幅 N_x，高さ N_y である画像輝度値の平均値 f_{ave} は数学的に以下の様に記述できる．

$$f_{ave} = \frac{1}{N_x N_y} \int_{N_y} \int_{N_x} f(x,y) dx dy. \tag{1}$$

さて，ここで画像 A の輝度値データを A_DATA として取り扱う．python ライブラリ Image を用いると，各画像の輝度値は A_DATA.getpixel((ii,jj)) として得ることが出来る．ここで，x 方向は ii，y 方向は jj によって配列のアドレスを参照する．また，画像の幅は A_DATA.size[0]，高さは A_DATA.size[1] として参照することが出来る．画像 A の平均輝度値は以下の演算によって求められる．

表 1　ビットマップ画像の配列

$f(0,4)$	$f(1,4)$	$f(2,4)$	$f(3,4)$	$f(4,4)$
$f(0,3)$	$f(1,3)$	$f(2,3)$	$f(3,3)$	$f(4,3)$
$f(0,2)$	$f(1,2)$	$f(2,2)$	$f(3,2)$	$f(4,2)$
$f(0,1)$	$f(1,1)$	$f(2,1)$	$f(3,1)$	$f(4,1)$
$f(0,0)$	$f(1,0)$	$f(2,0)$	$f(3,0)$	$f(4,0)$

Listing 1　sample01

```
ave = 0
for jj in range(A_hei):
⟶for ii in range(A_wid):
⟶⟶ave += float(A_DATA.getpixel((ii,jj)))
ave = float(ave/(A_wid*A_hei))
print("Average : " + str(ave))
```

6　レポート課題 B1

プログラム kadai_B1.py，12 行目のサブルーチン def calc_ave(A_DATA, A_wid, A_hei): の内部を書き換え，画像の平均輝度を計算するアルゴリズムを実装せよ．実装したプログラムを使用し，表 2 に列記した画像に対してそれぞれの平均輝度値を計算せよ．またその大きさ比較し，その原因を考えよ．

How to execute 01

```
$␣python␣kadai_B1.py ↵
$␣python␣kadai_B1.py ␣-i␣ 入力画像.bmp ↵
```

表 2　課題 B1　平均輝度値の比較

画像	平均値
rekidai-index-089-koizumi.bmp	
rekidai-index-090-abe.bmp	
rekidai-index-091-fukuda.bmp	
rekidai-index-092-aso.bmp	
rekidai-index-093-hatoyama.bmp	
rekidai-index-094-kan.bmp	
rekidai-index-095-noda.bmp	
rekidai-index-096-abe.bmp	
rekidai-index-099-suga.bmp	
rekidai-index-100-kishida.bmp	

7　移動平均（平滑化）

関数が連続な場合であってもノイズを含む場合や滑らかさが欠ける場合がある．このような場合，関数を平滑化する手段として移動平均値を用いることがある．移動平均値 $f_{ma}(x)$ は元関数 $f(x)$ に対して

$$f_{ma}(x) = \frac{1}{L}\int_{-L/2}^{L/2} f(x+\xi)d\xi, \tag{2}$$

のように定義される．ここで，L は移動平均に用いる窓の大きさであり，この大きさにより滑らかさが決まる．なお，関数の端部（開始部分と終了部分）は窓関数の大きさにより一意に評価できないので，工夫が必要となる．また，この移動平均を二次元に拡張すると以下のよう表される．

$$f_{ma}(x,y) = \frac{1}{L_xL_y}\int_{-L_x/2}^{L_x/2}\int_{-L_y/2}^{L_y/2} f(x+\xi,y+\eta)d\xi d\eta. \tag{3}$$

画像 A に対して，縦 len pixel，横 len pixel の窓サイズで移動平均を行う演算は以下のように記述できる．ただし，len は奇数であることに注意されたい．

Listing 2　sample02

```
hlen=int((len-1)/2)
for jj in range(A_hei):
→→for ii in range(A_wid):
→→→→ave = 0.0
→→→→num = 0
→→→→for tj in range(-hlen,hlen+1):
→→→→→→for ti in range(-hlen,hlen+1):
→→→→→→→→if  0 <= ii + ti < A_wid \
→→→→→→→→and 0 <= jj + tj < A_hei:
→→→→→→→→→→ave += float( \
→→→→→→→→→→A_DATA.getpixel((ii+ti,jj+tj)))
→→→→→→→→→→num += 1
→→→→ave = int(ave/num)
→→→→B_DATA.putpixel((ii,jj), ave)
```

8　レポート課題 B2

プログラム kadai_B2.py，12 行目のサブルーチン def calc_smooth(A_DATA, B_DATA, A_wid, A_hei, len): の内部を書き換え，画像の移動平均輝度を計算するアルゴリズムを実装し配列 B_DATA に結果を代入せよ．このプログラムは B_DATA を smooth.bmp（デフォルト）として出力する．画像 rekidai-index-100-kishida.bmp に対して移動平均を計算しその結果を表示せよ．窓の大きさを変化させ窓の大きさが結果に与える影響を考えよ．

How to execute 02

$␣python␣kadai_B2.py ↵
$␣python␣kadai_B2.py ␣-i␣ [入力画像.bmp]
␣-o␣ [出力画像.bmp] ␣-n␣ [平滑化窓サイズ（奇数）] ↵

9　相互相関係数

二つの異なる関数 $f(x)$ と $g(x)$ を比較し類似度を定量的に示す指数として，以下に示される相互相関係数が用いられる．

$$c = \frac{\frac{1}{N_x}\int_{N_x}\{f(x)-f_{ave}\}\{g(x)-g_{ave}\}dx}{\sqrt{\frac{1}{N_x}\int_{N_x}\{f(x)-f_{ave}\}^2dx}\sqrt{\frac{1}{N_x}\int_{N_x}\{g(x)-g_{ave}\}^2dx}}. \tag{4}$$

これを二次元に拡張すると以下のようになる．

$$c = \frac{a_3}{a_1a_2} \tag{5a}$$

$$\begin{cases} a_1 = \sqrt{\frac{1}{N_xN_y}\int_{N_x}\int_{N_y}\{f(x,y)-f_{ave}\}^2dxdy}, \\ a_2 = \sqrt{\frac{1}{N_xN_y}\int_{N_x}\int_{N_y}\{g(x,y)-g_{ave}\}^2dxdy}, \\ a_3 = \frac{1}{N_xN_y}\int_{N_x}\int_{N_y}\{f(x,y)-f_{ave}\}\{g(x,y)-g_{ave}\}dxdy. \end{cases} \tag{5b}$$

相互相関係数 c は -1 から 1 までの値となり，一般的に $c \geq 0.6$ の場合に相関があると言われる．相関がないとは相関係数が 0 であり，相関係数が -1 の時は逆相関と言う．

10　レポート課題 B3

プログラム kadai_B3.py，12 行目のサブルーチン def calc_crc(A_DATA, B_DATA, A_wid, A_hei): の内部を書き換え，画像 A_DATA と B_DATA を計算するアルゴリズムを実装せよ．以下の画像同士の直接相互相関係数を計算し，類似度を比較すること．また，類似度の差が生じる理由を考えよ．

- ① rekidai-index-089-koizumi
- ② rekidai-index-090-abe.bmp
- ③ rekidai-index-091-fukuda.bmp
- ④ rekidai-index-092-aso.bmp
- ⑤ rekidai-index-093-hatoyama.bmp
- ⑥ rekidai-index-094-kan.bmp
- ⑦ rekidai-index-095-noda.bmp
- ⑧ rekidai-index-096-abe.bmp
- ⑨ rekidai-index-099-suga.bmp.bmp
- ⑩ rekidai-index-100-kishida.bmp

How to execute 03

$␣python␣kadai_B3.py ↵
$␣python␣kadai_B3.py ␣-i1␣ [第一画像.bmp]
␣-i2␣ [第二画像.bmp] ↵

表 3　課題 B3　直接相互相関係数の比較

	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩
①										
②										
③										
④										
⑤										
⑥										
⑦										
⑧										
⑨										
⑩										

11　レポート課題 B4

プログラム kadai_B4.py，12 行目のサブルーチン def calc_matching(A_DATA, B_DATA, C_DATA, A_wid, A_hei, B_wid, B_hei, skip): の内部を書き換えテンプレート画像 B を用い，画像 A との相互相関関数を計算するアルゴリズムを実装せよ．実装したプログラムにて，テンプレート画像 cut_naikaku_r050913.bmp を用い，naikaku_r050913.bmp から相互相関係数に基づいた顔検出を行い，検出した座標 (x, y) を答えよ．何を基準に顔と判断すれば良いか．また，誤検出する場合はどのような場合か考え，その改善策を示せ．なお，プログラム kadai_B4.py は相互相関係数 c を規格化し C_DATA へ代入し，これを find.bmp（デフォルト）として出力する．dis_face.py を用いて結果を可視化し，顔検出の結果を示せ．

How to execute 04

```
$␣python␣kadai_B4.py ↵
$␣python␣kadai_B4.py ␣-i1␣ [ターゲット画像.bmp]
␣-i2␣ [テンプレート画像.bmp] ␣-o␣ [出力画像.bmp]
␣-n␣ [探索間隔（整数）] ↵
$␣python␣dis_face.py ↵
$␣python␣dis_face.py ␣ [ターゲット画像.bmp] ␣
[テンプレート画像.bmp] ␣ [出力画像.bmp] ␣
[x 方向検出位置（整数）] ␣ [y 方向検出位置（整数）] ↵
```

12　レポート課題 B5

プログラム kadai_B5.py，12 行目のサブルーチン def calc_vec(): の内部を書き換え，相互相関係数を用いた粒子画像速度計測法（PIV）[1, 2] を作成せよ．なお，出力結果を dis_vec.py を用いて可視化し，正解データ pivstandard_ans.csv と比較せよ．

How to execute 05

```
$␣python␣kadai_B5.py ↵
$␣python␣kadai_B5.py ␣-i1␣ [第一画像.bmp] ␣-i2␣
[第二画像.bmp] ␣-o␣ [出力ベクトルファイル名.csv] ␣-x␣
[x 方向探索間隔（整数）] ␣-y␣ [y 方向探索間隔（整数）]
␣-s␣ [探索窓サイズの半値（整数）] ␣-n␣
[検査窓サイズの半値（整数）] ↵
$␣python␣dis_vec.py ↵
$␣python␣dis_vec.py ␣ [出力ベクトルファイル名.csv] ↵
```

13　レポート課題 B6

実験室で撮影した画像を用い，課題 B5 で作成した自作した PIV により速度分布を得よ．実験画像には背景やセンサノイズの写り込みがある．まず，bgr.py を用いて背景除去を行いうこと．bgr.py は背景画像 bg_rootname.bmp と背景処理画像 bgr_rootname_6 桁連番.bmp を出力する，次に，imagej で背景画像を読み込む．imageJ のメインウィンドウから Image > Adjust > Brightness/Contrast を選択する．B&C ウィンドウのバーを用い，画像の明度を調整し輪郭を見やすくする．画像ファイル上でカーソルを動かすと，その座標位置がメインウィンドウ左下に表示される．翼の輪郭座標を shape.csv に書き込む．最後に，背景除去された画像に対して PIV を実行し，流動の時間変化を議論せよ．なお，出力結果を dis_vec2.py を用いて可視化すること．

How to execute 06

```
$␣python␣bgr.py ↵
$␣python␣bgr.py ␣-i␣ [入力画像のルート名]
␣-s␣ [入力画像の開始番号（整数）]
␣-n␣ [入力画像の数（整数）] ↵
$␣imagej␣ [背景画像.bmp]
$␣python␣dis_vec2.py ↵
$␣python␣dis_vec2.py ␣ [出力ベクトルファイル名.csv] ␣
[翼輪郭ファイル名.csv] ↵
```

参考文献

[1] 可視化情報学会 編，“PIV ハンドブック”，森北出版，(2002).
[2] 笠木伸英，木村龍治，西岡通男，日野幹雄，保原充 編，“流体実験ハンドブック”，朝倉書店，(1997).