情報デザイン専攻

画像情報処理論及び演習I

-デジタル画像の表現と応用-
画像処理プログラミングの基礎

第４回講義
水曜日１限
教室6218情報処理実習室

吉澤　信
shin@riken.jp, 非常勤講師
大妻女子大学　社会情報学部
独立行政法人 理化学研究所

今日の授業内容

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec04.pdf

今日はプログラミングをやります！

① レポートについて.
② 演習：入出力、2値化、多値化、Hue疑似カラー、ヒストグラム作成.

最初のレポートは↑の内容なので頑張ってくださいねー(^^;

第一回レポートについて、

www.riken.jp/brict/Yoshizawa/Lectures/Report01.doc

1. ↑のレポートをWindowsのWordか、LinuxのOpenOfficeで編集しPDF化.
2. ソースファイルや入出力画像と共にフォルダーに入れてzipファイルに圧縮し、zipファイルをwebから提出（〆切5/29）.
3. 作成方法の注意点と提出方法は↓を参照してください.

www.riken.jp/brict/Yoshizawa/Lectures/Report_ex.pdf

第一回レポートについて、

1. みなさん、デジカメや携帯で撮ったオリジナルの画像をレポートでは使ってください.
2. ppm, pgmへ変換するには、convertを使います.例えば、

「convert -quality 100 -compress none -comment "" input.bmp output.ppm」

3. bmpやjpgへ変換も同じで、

例えば、「convert -quality 100 input.ppm output.bmp」

Ex01.zipの内容

Ex01.zipの内容：

共用： SimpleImage.h

グレースケール画像用：
- pgmio.h
- ex01.cxx

カラー画像用：
- ppmio.h
- ex01_2.cxx

✓ 今後の全ての演習はこれらのファイル中のプログラム構造を雛形として使うので中身をよく見ておいてください.
✓ カラー画像用ではImageクラスをR,G,B3つ使っているだけです.

演習4-1: カラーからグレースケールへの変換

www.riken.jp/brict/Yoshizawa/Lectures/Ex01.zip

1. カラー画像(ppm)を読み込んでR,G,Bの平均値を輝度値とするグレースケール画像(pgm)を保存するプログラムを作成せよ.
2. argvを使って、入力ファイル名、出力ファイル名を指定出来る事.
3. ヒント：ex01.cxxとex01_2.cxx.
4. #include<stdlib.h>を忘れずに！

Shin Yoshizawa: shin@riken.jp

演習4-2: 閾値を用いた2値化

1. pgm画像を読み込み、閾値以下の輝度値を0、閾値以上の輝度値を255に変更した2値化画像(pgm)を作成・保存するプログラムを作成せよ.
2. argv, atoiを使って、入力ファイル名、出力ファイル名、閾値を指定出来る事.
3. ヒント：ex01_02.cxxのコメントアウト部分.

Shin Yoshizawa: shin@riken.jp

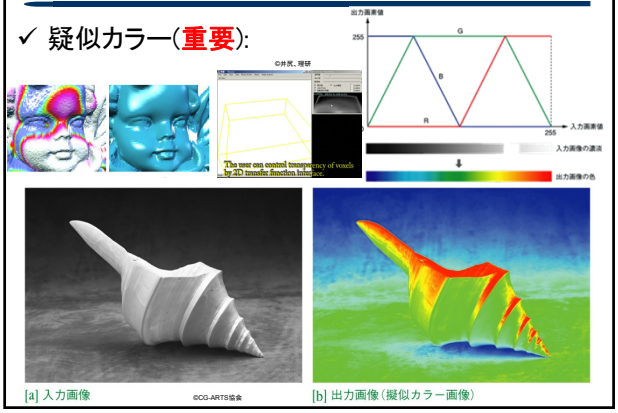
演習4-2: ヒント

1. lena.pgmで閾値を64、96、128、160、192で実行した結果は以下の様になります.

Shin Yoshizawa: shin@riken.jp

復習:トーンカーブ(Tone Reproduction Curve)12

✓ 疑似カラー(重要):



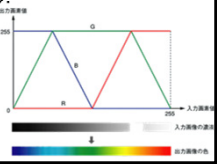
[a] 入力画像 ©CG-ARTS協会 [b] 出力画像(擬似カラー画像)

Shin Yoshizawa: shin@riken.jp

演習4-3: Hue変換

1. pgm画像を読み込んでHue疑似カラー画像へ変換するプログラムを作成せよ.
2. argvを使って、入力画像ファイル名、出力画像ファイル名を指定出来る事.
3. ヒント：入力の輝度値⇒HueのRGB変換用の関数を三つ用意する.

右のグラフと同様に色を変換する.



Shin Yoshizawa: shin@riken.jp

演習4-3: ヒント

$$\mathrm{Hue}R(x)=\begin{cases}0 & 0\le x<128\\ (255/64)x-510 & 128\le x<192\\ 255 & 192\le x\le 255\end{cases}$$

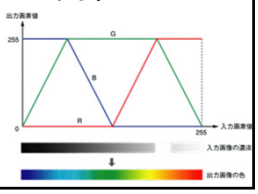
$$\mathrm{Hue}G(x)=\begin{cases}(255/64)x & 0\le x<64\\ 255 & 64\le x<192\\ -(85/21)x+(7225/7) & 192\le x\le 255\end{cases}$$

$$\mathrm{Hue}B(x)=\begin{cases}255 & 0\le x<64\\ -(255/64)x+510 & 64\le x<128\\ 0 & 128\le x\le 255\end{cases}$$

✓ forの二重ループで変換し保存.

✓ y= ax+ bの連立方程式を解くと左の関数が導出出来る.

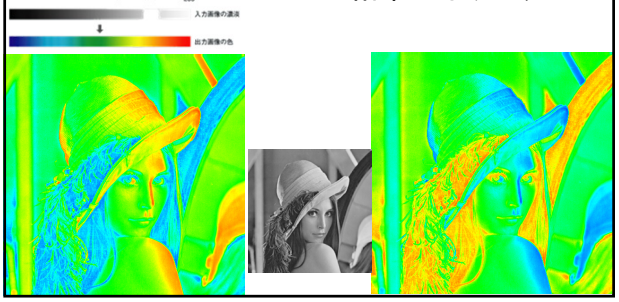
✓ 注意点：プログラム内で(255/64)などは浮動小数点(255.0/64.0)とする事.



Shin Yoshizawa: shin@riken.jp

演習4-3: ヒント

1. lena.pgmでそのままin->img[i][j]を変換したのが左、255.0-in->img[i][j]とネガポジ反転して変換したのが右の結果になります.



Shin Yoshizawa: shin@riken.jp

演習4-4: 統計

1. pgm画像を読み込んで輝度値の最大値、最小値、平均値、及び中央値を計算し表示するプログラムを作成せよ.
2. argvを使って、入力画像ファイル名を指定出来る事.
3. ヒント：中央値は、輝度値の値を大きさでsortした場合に、N/2番目の値. ただしNは画素数.

Shin Yoshizawa: shin@riken.jp

演習4-4: ヒント

1. 中央値は画像をImage *inとすると以下の様にstandard libraryを使うと簡単.

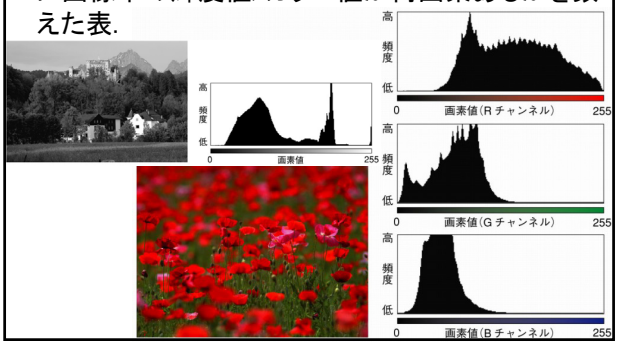
```
#include<algorithm>
#include<vector>
std::vector<double> val;
forの2重ループで
val.push_back(in->img[i][j]);
その後に
std::sort(val.begin(),val.end());
double median = val[val.size()/2];
で計算.
```

lena.pgmの正解は、
最大値: 245
最小値: 26
平均：124.604736…
中央値: 129

Shin Yoshizawa: shin@riken.jp

復習：ヒストグラム(Histogram)

✓ 画像の頻度表(ヒストグラム)とは量子化の階調毎に画像中の輝度値/カラー値が何画素あるかを数えた表.



Shin Yoshizawa: shin@riken.jp

演習4-5: ヒストグラム作成

1. pgm画像を読み込んで輝度値のヒストグラムを出力するプログラムを作成せよ.
2. argv, atoiを使って、入力画像ファイル名、出力ヒストグラムファイル名とビンの数を指定出来る事.
3. ヒント1：FILE *fp = fopen(出力ファイル名,"w"); fprintf(fp,"%d %ld¥n",ビンのID,頻度); fclose(fp);
4. ヒント2：int N = atoi(argv[3]);
   long *hist = new long[N]; delete [] hist;
5. 表示はxmgrace or gnuplot.

Shin Yoshizawa: shin@riken.jp

演習4-5: ヒント

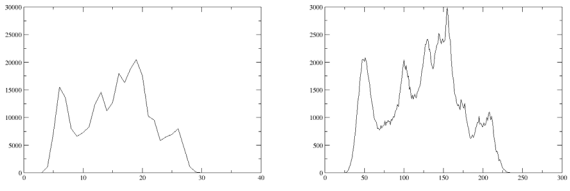
1. ビンの数N、ヒストグラムの配列をlong *hist, 入力画像をImage *inとすると、for(i=0;i<N;i++)hist[i]=0;の後にforの二重ループ(iとj)で以下を計算.

```
double val = (in->img[i][j])/((double)(in->gray));
val *= (N-1);
int vali = ((int)(val));
if(val-((double)(vali))>=0.5)vali++;
if(vali>=N)vali=N-1;
 hist[vali]++;
```

Shin Yoshizawa: shin@riken.jp

演習4-5: ヒント

1. lena.pgmの輝度値ヒストグラムです.ここでは、xmgraceを使ってグラフ化しました. ビンの数は32(左)と256(右)の場合です.



Shin Yoshizawa: shin@riken.jp

次回の予定

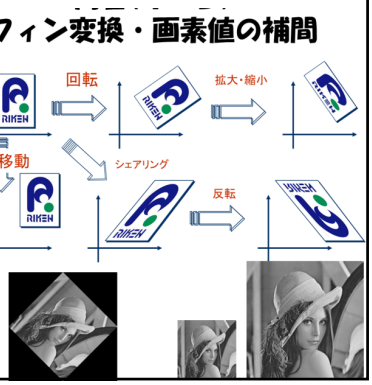
1回
2回
3回
4回
5回
6回
7回
8回
9回
10回
11回
12回
13回
14回
15回

基礎

アフィン変換・補間

領域抽出

画像合成



Shin Yoshizawa: shin@riken.jp

復習：参考資料

Shin Yoshizawa: shin@riken.jp

演習：Ex01.zip

1. www.riken.jp/brict/Yoshizawa/Lectures/Ex01.zipをダウンロードして展開してください.

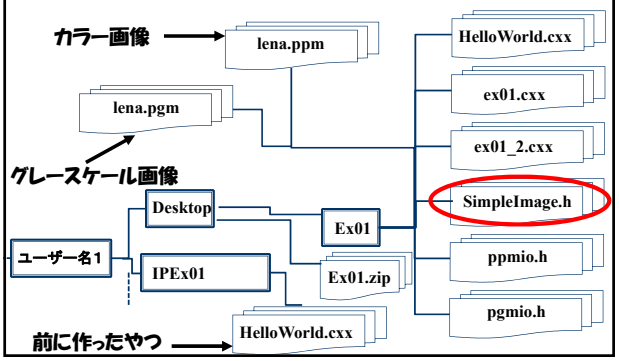
Firefox:編集→設定→一般→ダウンロード→「ファイルごとに保存先を指定する」にチェックを入れてください.

2. 端末で「g++ ex01.cxx」として実行ファイルa.outを作成後に「./a.out lena.pgm test.pgm」としてください。その後に「display test.pgm &」と「display lena.pgm &」を実行して同じ画像である事を確認してください.
3. 同様に「g++ ex01_2.cxx」、「./a.out lena.ppm test.ppm」、「display lena.ppm &」、「display test.ppm &」として同じカラー画像である事を確認してください.

Shin Yoshizawa: shin@riken.jp

Ex01.zipの内容

SimpleImage.hは画像クラスImageが記述されています.



Shin Yoshizawa: shin@riken.jp

C++クラスの基礎

```
class クラス名{   /* 設計図の様なものでクラス=新しい型 */
    public:   /* パブリックの場合は、クラスの外から参照可能 */
    メンバー変数   /* クラスが持っている変数、構造体、クラス内クラス */
    クラス名(){   /* コンストラクター：newされたときに呼ばれる. */
    }
    クラス名(引数){} /* コンストラクターは複数あってよい */

    ~クラス名(){   /* デコンストラクター：delete されたときに呼ばれる. */
    }
    戻り値 メソッド名(引数){} /* メソッドを作れる= */
    private: /* プライベートの場合は、クラスの外から参照不可 */
};
```

Shin Yoshizawa: shin@riken.jp

多重ポインターから多次元配列を作る方法

✓ 1重ポインターから1次元配列を作る方法：

```
double *AAA = new double[N];
```

これで、A[0], A[1], …A[N−1]まで配列として使える.

- 使い終わったらメモリの開放が必要：`delete [] AAA;`

✓ 2重ポインターから2次元配列を作る方法：

```
double **AAA = new double *[N];
for(int i=0;i<N;i++)AAA[i] = new double[M];
```

これで、A[0][0], A[0][1], …A[0][M−1], A[1][0], A[1][1],…A[N−1][M−1]まで配列として使える.

- 使い終わったらメモリの開放が必要：

```
for(int i=0;i<N;i++) delete [] AAA[i];
delete [] AAA;
```

Shin Yoshizawa: shin@riken.jp

Imageクラス

SimpleImage.h: 2次元配列で一色の画像を表すImageクラス.

#include"SimpleImage.h"した後の使い方例:

宣言・メモリ確保 (allocation):

```
Image *in = new Image();
Image *out = new Image(in->sx,in->sy);
```

```
getPGM(in,argv[1]);
```

処理:

```
/* Only Copy */
for(i=0;i<in->sy;i++)
 for(j=0;j<in->sx;j++){
   out->img[i][j] = in->img[i][j];
 }
```

- ✓ 画像サイズ:縦:sy、横sx.
- ✓ (座標(i,j)での)画素値: img[i][j]

メモリの開放:

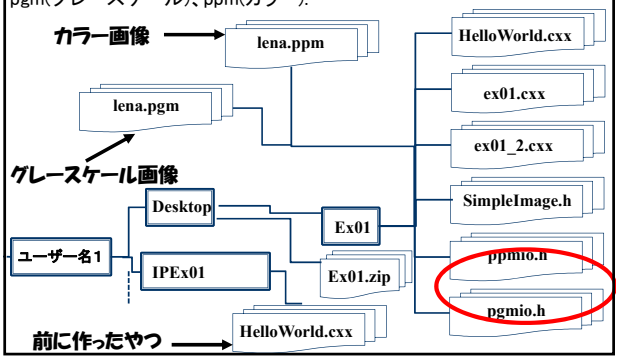
```
delete out;
delete in;
```

SimpleImage.h

Shin Yoshizawa: shin@riken.jp

Ex01.zipの内容

pnm画像の入出力関数が記述されています.
pgm(グレースケール)、ppm(カラー).

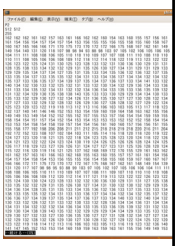


Shin Yoshizawa: shin@riken.jp

pnm画像フォーマット

- ✓ 一番簡単な画像フォーマットです:
  - グレースケール画像は「.pgm」、カラー画像は「.ppm」でテキスト形式とバイナリー形式があります.
  - グレースケール(.pgm):

1行名: テキストで「P2」
2行目: 画像サイズ(横:width 縦:height)
3行目:画素の階調(最大値) 8bitの場合は255
4行目から: integerで画素値スペース画素値…



  - カラー(.ppm):

1行名: テキストで「P3」
2行目: 画像サイズ(横:width 縦:height)
3行目:画素の階調(最大値) 8bitの場合は255
4行目から: integerでR G B R G B R G B…

Shin Yoshizawa: shin@riken.jp

pgmio.h:getPGM(), savePGM()

pgmio.h: pgmファイルの入出力を行う2つの関数.

画像入力:

```
void getPGM(Image *in,char *filename)
```

画像出力:

```
void savePGM(Image *in,char *filename)
```

#include"pgmio.h"した後の使い方例:

入力:

```
getPGM(in,argv[1]);
```

- ✓ argv[1]で渡されたファイル名のpgm画像を開いてImageクラスinに入れる.
- ✓ 注意:inは下記の様に画像サイズなしでnewされていないといけない!

```
Image *in = new Image();
```

出力:

```
savePGM(out,argv[2]);
```

- ✓ argv[2]で渡されたファイル名にoutの中身をpgm画像として保存.
- ✓ 注意:outは下記の様に画像サイズありでnewされていないといけない!

```
Image *out = new Image(in->sx,in->sy);
```

pgmio.h

Shin Yoshizawa: shin@riken.jp

ppmio.h:getPPM(), savePPM()

- ✓ 同様に、カラー画像は、#include"ppmio.h"の後で、getPPM()とsavePPM()を用いてppm画像の入出力が出来ます.
  - void getPPM(Image *R, Image *G, Image *B, char *filename)
  - void savePPM(Image *R, Image *G, Image *B, char *filename)
- ✓ pgmio.hと同様に、getPPM()を使う場合に変数R,G,Bは、以下の様にnewされている必要があります.

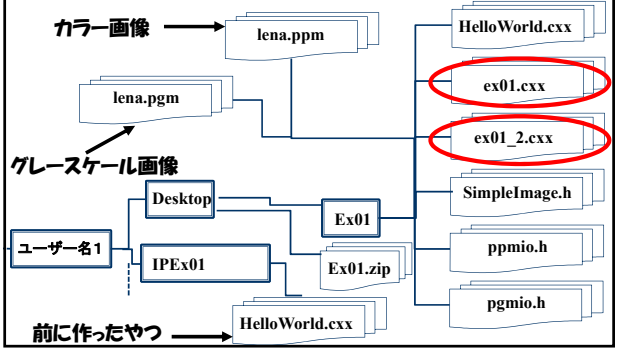
```
Image *R = new Image();
Image *G = new Image();
Image *B = new Image();
```

- ✓ delete R; delete G; delete B;でメモリ開放.

Shin Yoshizawa: shin@riken.jp

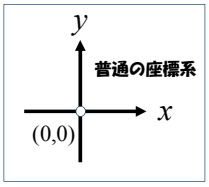
Ex01.zipの内容

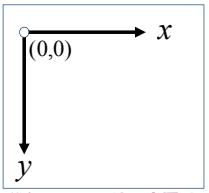
ex01.cxxはpgm画像を読み込んでそのままセーブするプログラム.
ex01_2.cxxはppm画像を読み込んでそのままセーブするプログラム.



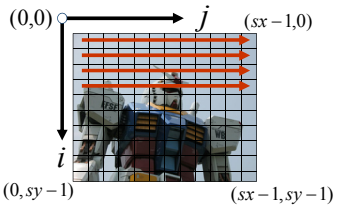
Shin Yoshizawa: shin@riken.jp

復習:デジタル画像の座標と配列





画像処理でよく使う座標系



輝度値の配列表現：

int $I[sy][sx]$;　$for(i=0;i<sy;i++){}$

double $I[sy][sx]$;　$for(j=0;j<sx;j++){}$

$I[i][j]=\ldots$

}

}

Shin Yoshizawa: shin@riken.jp

pgm入出力

ex01.cxxはpgm画像を読み込んでそのままセーブするプログラムです.

```
#include<stdio.h>
#include<math.h>
#include"SimpleImage.h"
#include"pgmio.h"
int main(int argc,char *argv[]){
  if(argc!=3){
    printf("Please use as ./a.out input.pgm output.pgm\n");
    return 0;
  }
  Image *in = new Image();
  getPGM(in,argv[1]);
  Image *out = new Image(in->sx,in->sy);
  /* main processing */
  int x,y;
  /* Only Copy */
   for(i=0;i<in->sy;i++)
    for(j=0;j<in->sx;j++){
      out->img[i][j] = in->img[i][j];
    }
   /* End Copy */

  /*
  double threshold = 128.0;
  for(i=0;i<in->sy;i++)
    for(j=0;j<in->sx;j++){
      if(in->img[i][j]>=threshold){
        out->img[i][j] = in->img[i][j];
      }else{
        out->img[i][j] = 0.0;
      }
    }
   */
  /* end main processing */

   savePGM(out,argv[2]);

  delete out;
  delete in;

  return 0;
}
```

入力用Imageクラスinの宣言・new.

argv[1]で渡されたファイル名のpgm画像を開いてImageクラスinに入れる.

出力用Imageクラスoutの宣言・new.

Inからoutへ画素の値をコピー.

SimpleImage.h

✓ SimpleImage.h: 2次元配列で一色(グレースケール)の画像を表すImageクラス

argv[2]で渡されたファイル名にoutの中身をpgm画像として保存.

pgmio.h

✓ pgmio.h: pgmファイルの入出力を行う2つの関数.

In、out領域の開放(delete).