情報デザイン専攻

画像情報処理論及び演習I

-補習・復習・予習-
レポート1～3回のQ9 & Q10
エンボス画像、勾配強度、Gaussian & Laplacianフィルタ

第14回講義
水曜日１限
教室6218

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

Shin Yoshizawa: shin@riken.jp

今日の授業内容

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec11.pdf

① みんな来るまで…9:40頃まで
- ✓ レポート第４回の採点結果を取りに来てください.
- ✓ (今日までの)自分の成績が知りたい人は教えます.

② レポート1～3までのQ9とQ10の解説＆レポート第1～4回の質問.

③ ↑出来ちゃってる人は…次ページのプログラミング課題をやってください. 出来たら成績に加点します！

みなさん良く頑張りましたd(>_・)
今日で通常の授業は終わりです.

- ✓ 単位ヤバイ人と1段上の評価欲しい人:は**補講日(7/29:5限)**に来てください.

みんな最後まで来てくれてありがとーo(≧▽≦)o

Shin Yoshizawa: shin@riken.jp

レポート出来ちゃってる人への課題

- ✓ 以下のプログラムを一から作ってみましょう！
- ✓ 出来たら手を挙げて呼んでください、1プログラムにつき最終成績に2.5点加点(レポート点約17点分・出席約0.9回分)します.
- ✓ ヒントは本講義資料の一番後ろにあります.
  - エンボス画像生成.
  - 勾配強度画像生成.
  - Gaussianフィルタ.
  - Laplacianフィルタ.

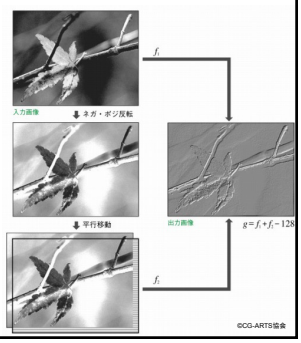
理論は後期にやります. 後期のレポートで出します！

Shin Yoshizawa: shin@riken.jp

出来る人のための課題1

✓ エンボス画像生成↓のプログラムを一から作成！

1. pgm画像を読み込む、画像Aとする.
2. ネガポジ反転し画像Bとする.
3. Bを平行移動しAと合成する.
4. 結果を0～255に正規化しpgmでセーブ.



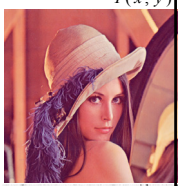
Shin Yoshizawa: shin@riken.jp

出来る人のための課題2

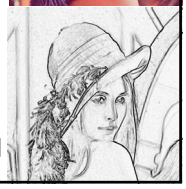
✓ 勾配強度画像生成↓のプログラムを一から作成！

1. pgm画像を読み込む、画像Aとする.
2. Aからx,y方向の微分を差分近似し画像B,C(勾配ベクトル画像)とする.
3. BとCから勾配ベクトルの大きさを計算し画像Dとする
4. Dを0～255に正規化しpgmでセーブ.

$I(x,y)$



入力



$$\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$$

$\|\nabla I(x,y)\|$ 勾配強度画像

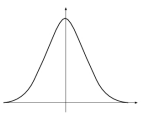
Shin Yoshizawa: shin@riken.jp

出来る人のための課題3

✓ Gaussianフィルタ↓のプログラムを一から作成！

連続式:

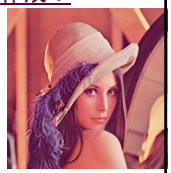
$$I^{new}(\mathbf{x}) = \frac{\int_{\Omega} g_\sigma(|\mathbf{x}-\mathbf{y}|)I(\mathbf{y})d\mathbf{y}}{\int_{\Omega} g_\sigma(|\mathbf{x}-\mathbf{y}|)d\mathbf{y}},$$



ガウス関数

$$g(r) = \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}}$$

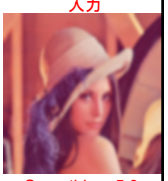
Smoothingパラメータ σ



入力

離散化式:

$$I^{new}(i,j) = \frac{\sum_{y=-r}^{y=r} g(i-y)(\sum_{x=-r}^{x=r} g(j-x)I(i-x,j-y))}{\sum_{y=-r}^{y=r} g(i-y)(\sum_{x=-r}^{x=r} g(j-x))},$$



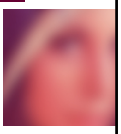
重み付平均の半径 r

Smoothing, 5.0

Shin Yoshizawa: shin@riken.jp

出来る人のための課題4

✓ Laplacianフィルタ↓のプログラムを一から作成！

連続式(拡散方程式):

$$\frac{\partial I(\mathbf{x},t)}{\partial t} = \Delta I(\mathbf{x},t),$$

離散式(拡散方程式の陽的前進一次差分近似):

$$I^{n+1}(i,j) = I^{n}(i,j) + \varepsilon(-9I(i,j) + \sum_{y=-1}^{y=1}\sum_{x=-1}^{x=1} I(i+y,j+x))/8,$$

ステップサイズパラメータ $\varepsilon < 0.5$

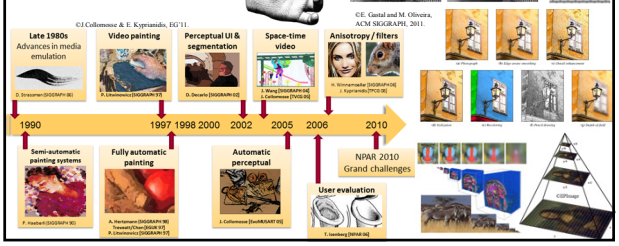
m回繰り返し適用する.

Shin Yoshizawa: shin@riken.jp

後期の予定1

「Artistic Stylization: ブラシ・エフェクト等の画像処理関連NPR: Non-Photorealistic Rendering」
「フィルタリング・ノイズ除去・画像復元」
「圧縮・周波数分解・多重解像度解析」
「HDRI・計算Photography」

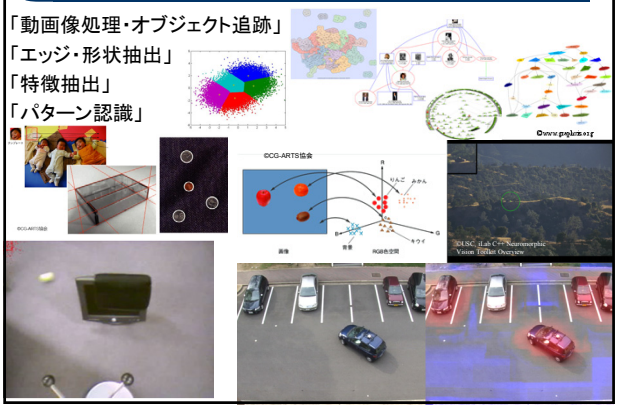




Shin Yoshizawa: shin@riken.jp

後期の予定2

「動画像処理・オブジェクト追跡」
「エッジ・形状抽出」
「特徴抽出」
「パターン認識」



Shin Yoshizawa: shin@riken.jp

補講日について

7月29日(金):5限16:20-17:50、6218教室.

✓ 補講対象:単位取得がヤバい人+「あと数点で一つ上の評価(C→BやA→S等)なので何とか...」という人なので、今日までの評価で満足(*^.^*)な人は来ても、来なくてもOK.

✓ 補講内容:

- 単位取得についての相談: 就職決まってるので何とかして！p(≧口≦)qという人は内定書や採用通知のコピーを持って来る事.
- 基本はレポート1～3のQ9&Q10とレポート4のQ4～Q8をやってもらいます.
- 今日と同じ「出来る人へのプログラミング課題」.

Shin Yoshizawa: shin@riken.jp

では、演習を始めてください

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec11.pdf

① みんな来るまで…9:40頃まで
  ✓ レポート第4回の採点結果を取りに来てください.
  ✓ (今日までの)自分の成績が知りたい人は教えます.
② レポート1～3までのQ9とQ10の解説&レポート第1～4回の質問.
③ ↑出来ちゃってる人は…プログラミング課題をやってください. フィルタは後期のレポートで出します！

みなさん良く頑張りましたd(>_・)
今日で通常の授業は終わりです.

✓ 単位ヤバい人と1段上の評価欲しい人:は補講日(7/29:5限)に来てください.

みんな最後まで来てくれてありがとーo(≧▽≦)o

Shin Yoshizawa: shin@riken.jp

レポート第1回Q9&Q10

Shin Yoshizawa: shin@riken.jp

レポート1－Q9

「カラー画像を逆変換(3次補間法)で拡大・縮小(Scaling)するプログラムを作成し、縦・横共に0.3倍、0.8倍、2.5倍、縦2倍・横0.5倍、縦0.3倍・横2.5倍した5つの画像を載せよ.」

Q9のヒント: Ex02.zip中のex02_3.cxxを改造すれば出来る.
また、
#include<stdlib>
と
double x_factor = atof(argv[3]);
double y_factor = atof(argv[4]);
を倍率の取得に使う.

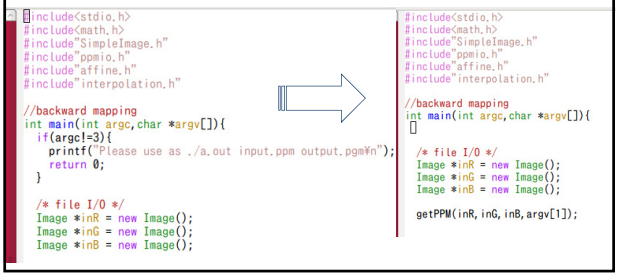
✓ 拡大縮小: Lec02_ex02.pdfの3ページ上段二つのスライド参照.
✓ 逆変換(3次補間法): Lec02_ex02.pdfの8ページ下段二つのスライド参照.

Shin Yoshizawa: shin@riken.jp

レポート1－Q9:2

✓ まずex02_3.cxxをemacsで開く.

1 : if(argc!=3){…}は(引数の数を増やすため)いらないので消す.

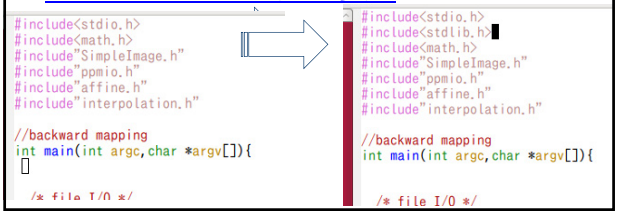


Shin Yoshizawa: shin@riken.jp

レポート1－Q9:3

2 : 引数を拡大・縮小率に使いたいので、文字列から数値に変換する関数atof()を使うためstdlib.hをincludeする.

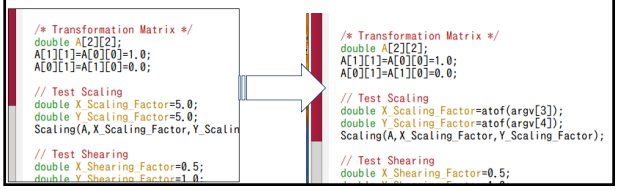
✓ 引数、argc,argv[]とatof(),atoi()はLec01_2.pdf、9ページの二つのスライド参照.



Shin Yoshizawa: shin@riken.jp

レポート1－Q9:4

3 : 入力・出力画像ファイル名で既に引数を二つ使っているので、第3引数argv[3]と第4引数argv[4]を文字列→doubleへ変換する関数atof()の中で呼び出し、x、y方向それぞれの拡大・縮小率に代入する.



Shin Yoshizawa: shin@riken.jp

レポート1－Q9:5

4 :

1. セーブ(Report01_Q9.cxx)する.
2. コンパイルする: g++ Report01_Q9.cxx -lm
3. 実行は新しく二つの引数を増やしたので、

./a.out 入力ppm画像 出力ppm画像 x方向倍率 y方向倍率
例えば、
./a.out lena.ppm test.ppm 2.0 0.3

縦・横共に0.3倍、0.8倍、2.5倍、縦2倍・横0.5倍、縦0.3倍・横2.5倍した5つの画像を計算する.

Shin Yoshizawa: shin@riken.jp

レポート1－Q10

「カラー画像を逆変換(3次補間法)で回転するプログラムを作成し、20度、45度、60度、90度、135度回転した5つの画像を載せよ、ただし回転中心は画像の中心とする」

Q10のヒント: Ex02.zip中のex02_3.cxxを改造すれば出来る.
また、
#include<stdlib>
と
double angle = atof(argv[3]);
を角度の取得に使う.

✓ 回転: Lec02_ex02.pdfの4ページ左下のスライド参照.
✓ 逆変換＆中心での回転:
✓ Lec02_ex02.pdfの7ページ中段左のスライド参照.
✓ 逆変換(3次補間法): Lec02_ex02.pdfの8ページ下段二つのスライド参照.

Shin Yoshizawa: shin@riken.jp

レポート1－Q10: 2

✓ まずex02_3.cxxをemacsで開く.

1 : if(argc!=3){…}は(引数の数を増やすため)いらないので消す.

2 : 引数を拡大・縮小率に使いたいので、文字列から数値に変換する関数atof()を使うためstdlib.hをincludeする.

ここまでは、Q9とまったく同じ！

Shin Yoshizawa: shin@riken.jp

レポート1－Q10: 3

3 : 回転行列をAに代入したいので、拡大・縮小をAに代入しているScaling(…)をコメントアウトし、Rotation(…)の行のコメントアウトを外す.

```
// Test Scaling
double X_Scaling_Factor=5.0;
double Y_Scaling_Factor=5.0;
Scaling(A,X_Scaling_Factor,Y_Scalin

// Test Shearing
double X_Shearing_Factor=0.5;
double Y_Shearing_Factor=1.0;
//Shearing(A,X_Shearing_Factor,Y_Sh

// Test Rotation
double Rotation_Angle = 45.0;
//Rotation(A,Rotation_Angle);
```

```
// Test Scaling
double X_Scaling_Factor=5.0;
double Y_Scaling_Factor=5.0;
//Scaling(A,X_Scaling_Factor,Y_Sc

// Test Shearing
double X_Shearing_Factor=0.5;
double Y_Shearing_Factor=1.0;
//Shearing(A,X_Shearing_Factor,Y_

// Test Rotation
double Rotation_Angle = 45.0;
Rotation(A,Rotation_Angle);
```

Shin Yoshizawa: shin@riken.jp

レポート1－Q10: 4

4 : Q9の拡大・縮小率と同じように回転角度に第３引数argv[3]をatof()関数から呼び出す.

5: Q9と同様にセーブ＆コンパイル＆実行.

ただし、Report01_Q10.cxx実行は

./a.out 入力ppm画像 出力ppm画像 回転角度

例えば、

./a.out lena.ppm test.ppm 45.0

```
//Shearing(A,X_Shearing_Factor,Y_

// Test Rotation
double Rotation_Angle = 45.0;
Rotation(A,Rotation_Angle);
```

```
// Test Rotation
double Rotation_Angle = atof(argv[3]);
Rotation(A,Rotation_Angle);
```

Shin Yoshizawa: shin@riken.jp

レポート第2回Q9＆Q10

Shin Yoshizawa: shin@riken.jp

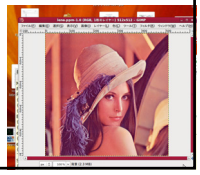
レポート2－Q9

「gimpを用いてトーンカーブを変更し入力画像をポスタリゼーションせよ」

Q9のヒント: 端末にて「gimp &」と打ち込みエンターキーでgimpが立ち上がるので、画像を開いて色メニューのトーンカーブを選ぶとトーンカーブの調整が可能.

1: 端末にて「gimp &」と打ち込みエンターキーでgimpを立ち上げる.

2: メニューバーのファイル->開く、で画像を読み込む.

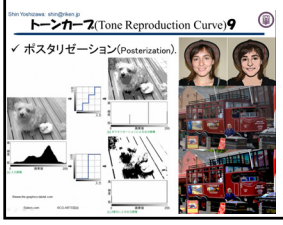


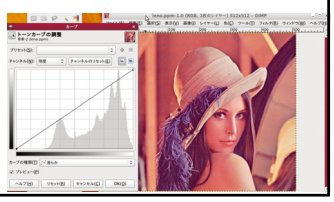
Shin Yoshizawa: shin@riken.jp

レポート2－Q9: 2

3: 色ツールのトーンカーブを選ぶ.

4: ポスタリゼーションは多値化(Lec03.pdfのページ１２、左上スライド参照)なので階段型のトーンカーブを生成し、画像を保存する.

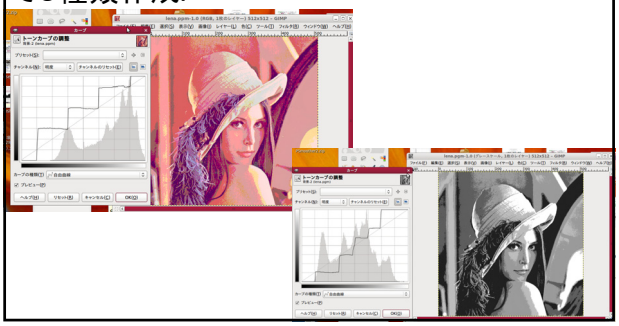




Shin Yoshizawa: shin@riken.jp

レポート2－Q9: 3

5: カラー画像とグレースケールの例を合わせて5種類作成.



Shin Yoshizawa: shin@riken.jp

レポート2－Q10

「gimpを用いてトーンカーブを変更し入力画像をソラリゼーションせよ」

1: Q9と同様に、端末にて「gimp &」と打ち込みエンターキーでgimpを立ち上げる.

2: メニューバーのファイル->開く、で画像を読み込む.

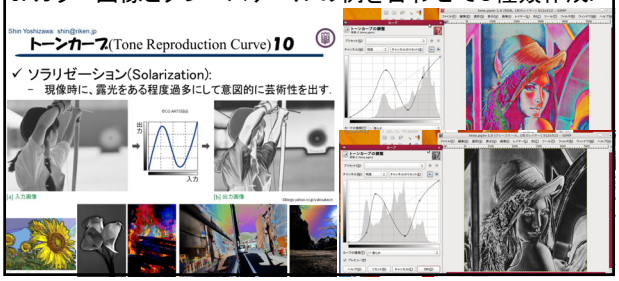
3: 色ツールのトーンカーブを選ぶ.

ここまでは、Q9とまったく同じ！

Shin Yoshizawa: shin@riken.jp

レポート2－Q10: 2

4: ソラリゼーションは露光過多(トーンカーブは一度上がって下がる、Lec03.pdfページ12右上スライド参照)なので山の次に谷になるトーンカーブを生成し、画像を保存する.
5: カラー画像とグレースケールの例を合わせて5種類作成.



Shin Yoshizawa: shin@riken.jp

レポート第3回Q9&Q10

Shin Yoshizawa: shin@riken.jp

レポート3－Q9

「閾値を与えてカラー画像を二値化するプログラムを作成し、閾値が64、96、128、160、192の場合の実行結果画像を載せよ」
Q9のヒント:Ex01/ex01_2.cxxにある様に、ppm画像を開いてR、G、Bの値を足して3で割った数が閾値未満なら出力にゼロ、以上なら255を入れる.

```
double tmp = (R->img[i][j]+G->img[i][j]+B->img[i][j]) /3;
if(tmp<threshold){  out->img[i][j] =255.0;
}else{  out->img[i][j]=0.0;
}
```

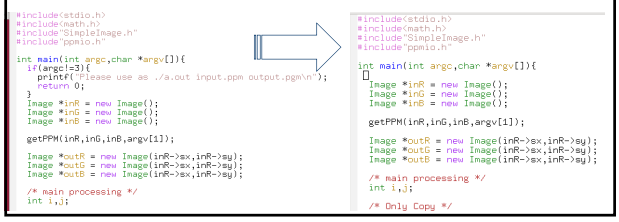
pgm画像のセーブはEx01/ex01.cxxを参照.
thresholdは#include<stdlib.h>, atoi()又はatof()を使ってargv[]から入力.

Shin Yoshizawa: shin@riken.jp

レポート3－Q9: 2

1: まず、Ex01/ex01_2.cxxをemacsで開く.

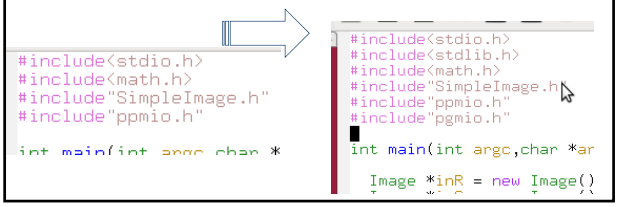
2: レポート1のQ9と同じようにif(argc!=3){…}は(引数の数を増やすため)いらないので消す.



Shin Yoshizawa: shin@riken.jp

レポート3－Q9: 3

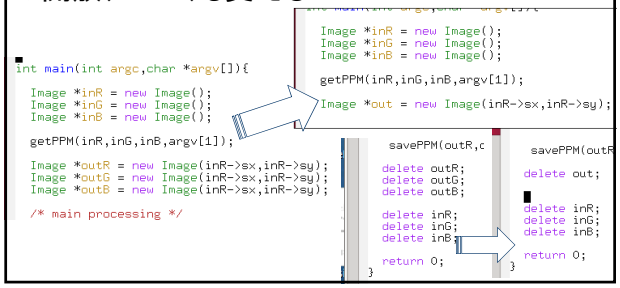
3:レポート1Q9と同様に、引数を閾値に使いたいので、文字列から数値に変換する関数atof()を使うためstdlib.hをincludeする.
4: また、グレースケール(pgm)画像を出力したいため、pgmio.hをincludeする.



Shin Yoshizawa: shin@riken.jp

レポート3－Q9: 4

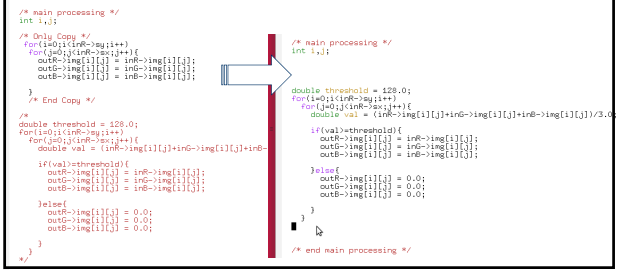
5:カラー画像出力用の宣言とメモリ確保をグレースケール用に変える. また、対応するメモリの開放(delete)も変える.



Shin Yoshizawa: shin@riken.jp

レポート3－Q9: 5

6: コピーしている場所はいらないので消す.
7: その下の閾値でカラー画像を変えている箇所のコメントアウトを外す.

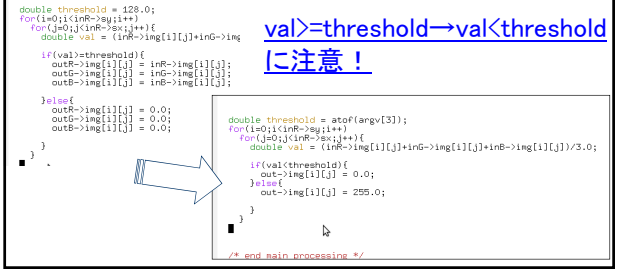


Shin Yoshizawa: shin@riken.jp

レポート3－Q9: 6

8: 閾値に128.0を代入している所を第3引数argv[3]をatof()関数から呼び出す.
9: その下をグレースケール用に変える.

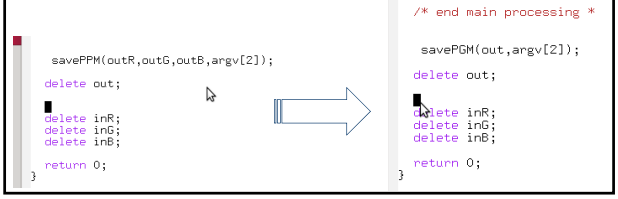
val>=threshold→val<threshold に注意！



Shin Yoshizawa: shin@riken.jp

レポート3－Q9: 7

10: カラー画像セーブ(ppm、savePPM(…))だった所をグレースケール画像セーブ(pgm、savepgm(…))に書き換える.
11: プログラムをセーブ(Report03_Q9.cxx).
12: コンパイル: g++ Report03_Q9.cxx -lm



Shin Yoshizawa: shin@riken.jp

レポート3－Q9: 8

13: 実行は、
./a.out 入力ppm画像 出力pgm画像 閾値

例えば、

./a.out lena.ppm test_128.pgm 128.0

閾値が64、96、128、160、192の場合の二値化画像を計算する.

Shin Yoshizawa: shin@riken.jp

レポート3－Q10(一番簡単！)

「Ex04.zip内のRun_ex04_2.shを(各自のオリジナルの画像で)実行した結果を載せて、それぞれの結果について考察を述べよ」

1: Ex04でRun_ex04_2.shを動かすだけ！！！

sh Run_ex04_2.sh 入力pgm画像 小領域の最小値(int)

例えば、

sh Run_ex04_2.sh lena.pgm 200

- ✓ 二値化、ラベリング、細線化：Lec05.pdf参照.
- ✓ Run_ex04_2.sh: Lec05.pdfのページ5右下のスライド参照.

Shin Yoshizawa: shin@riken.jp

補講日について

7月29日(金)：5限16:20−17:50、6218教室.

- ✓ 補講対象：単位取得がヤバィ人＋「あと数点で一つ上の評価(C→BやA→S等)なので何とか…」という人なので、今日までの評価で満足(*^.^*)な人は来ても、来なくてもOK.
- ✓ 補講内容：
  - 単位取得についての相談: 就職決まってるので何とかして！p(≧口≦)qという人は内定書や採用通知のコピーを持って来る事.
  - 基本はレポート1～3のQ9&Q10とレポート4のQ4～Q7をやってもらいます.
  - 今日と同じ「出来る人へのプログラミング課題」.

Shin Yoshizawa: shin@riken.jp

おわりに、

みなさん良く頑張りましたd(>_・)

今日で通常の授業は終わりです.

みんな最後まで来てくれてありがとー

o(≧▽≦)o

また後期の授業でお会いしましょう！

ヾ(^-^)ゞ

補講日(7/29:5限)来る方は、また今週末♪

Shin Yoshizawa: shin@riken.jp

みなさんへのメッセージ

- ✓ 初めて習うんだから、わからないのは当たり前だし、最初から出来なくてもOKです、でも、ずーと復習・自習や質問をしないで「わからない、難しい」となるのは違います→積極的に質問＆授業内容の復習をしましょう！
- ✓ あきらめずにTryする事が大事(^^;
- ✓ プログラミングって気合(笑)でなんとかなります.
  私も大学2年の前期までほとんど書けませんでしたが、半年ぐらいメチャやったら(ほぼ)何でも書ける様になりましたし、自転車と同じで一度書ける様になったら、それ以後は忘れないし簡単になります. 練習と思って他の授業のレポートでも(求められてなくても)プログラミングをしてみて、その結果をレポートに書いてみてください(凄く評価されるはず).
- ✓ 本講義でレポートを雛形から決まった形式で作成してもらったのは、みなさんが社会に出たときのためです.
  報告書、企画書や技術書などの作成は相手に伝わる文章を書ける様に訓練しないと、なかなか良い物が書けません→他の授業でも(求められなくても)WordやPDFの整った文章＆形式(段落・章立て等)でレポート作成を試みてください.

Shin Yoshizawa: shin@riken.jp

プログラミング課題のヒント

Shin Yoshizawa: shin@riken.jp

出来る人のための課題1

- ✓ エンボス画像生成↓のプログラムを一から作成！

1. pgm画像を読み込む、画像Aとする.
2. ネガポジ反転し画像Bとする:

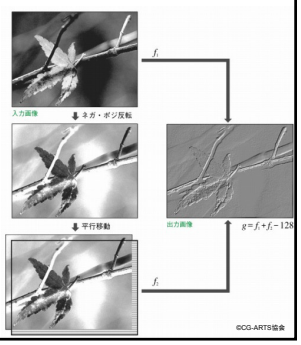
```
B->img[i][j]=255.0-A->img[i][j];
```

3. Bを平行移動しAと合成する:

```
C->img[i][j] = B->img[i+t][j+t]+A->img[i][j]-128.0;
```

4. 0～255に正規化しpgmでセーブ:

```
out->img[i][j]=255.0*(C->img[i][j]-min(C))/fabs(max(C)-min(C));
```



Shin Yoshizawa: shin@riken.jp

エンボス画像生成ヒント

1. pgm画像を読み込む:
   1. SimpleImage.hをincludeし入力画像用にメモリ確保を行う:
      Image *A = new Image();
   2. pgmio.hをincludeしgetPGM(Image *,char *)を使う.
2. ネガポジ反転し画像Bとする:
   1. 画像BをAと同じサイズで確保する:
      Image *B = new Image(A->sx,A->sy);
   2. forの二重ループ(0<=i<A->sy, 0<=j<A->sx)でBの中身(輝度値)を作る: B->img[i][j]=255.0-A->img[i][j];
3. Bを平行移動しAと合成する:
   1. 画像CをAと同じサイズで確保する:
      Image *C = new Image(A->sx,A->sy);
   2. forの二重ループ(0<=**i<A->sy-1**, 0<=**A->sx-1**)でCの中身を作る: C->img[i][j] = B->img[i+t][j+t]+A->img[i][j]-128.0;

Shin Yoshizawa: shin@riken.jp

エンボス画像生成ヒント2

4. 0～255に正規化しpgmでセーブ:
   1. 出力用のoutをAと同じサイズで確保する:
      Image *out = new Image(A->sx,A->sy);
   2. Cの輝度値の最小と最大を計算する:

```
double max,min;
max=min=C->img[0][0];
for(i=0;i<A->sy;i++)
  for(j=0;j<A->sx;j++){
    if(max<C->img[i][j])max=C->img[i][j];
    if(min>C->img[i][j])min=C->img[i][j];
  }
```

   3. forの二重ループでoutの中身を作る:

```
out->img[i][j]=255.0*(C->img[i][j]-min)/fabs(max-min);
```

   4. savePGM(Image*, char *)を使ってセーブする.
   5. newしたクラスはdeleteする事: delete A; delete B; delete C; delete out;

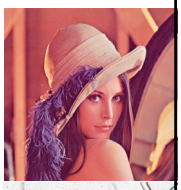
Shin Yoshizawa: shin@riken.jp

出来る人のための課題2

✓ 勾配強度画像生成↓の[プログラムを一から作成！]

1. pgm画像を読み込む、画像Aとする.
2. x,y方向の微分を差分近似し画像B,Cとする:

```
B->img[i][j]=A->img[i][j+1]-A->img[i][j];
C->img[i][j]=A->img[i+1][j]-A->img[i][j];
```



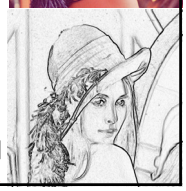
$I(x,y)$ 入力

3. 勾配ベクトルの大きさをDとする:

```
D->img[i][j] = sqrt(B->img[i][j]*B->img[i][j] +
A->img[i][j]*A->img[i][j]);
```

4. 0～255に正規化しpgmでセーブ:

```
out->img[i][j]=255.0*(D->img[i][j]-
min(D))/fabs(max(D)-min(D));
```



$$\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$$

$\|\nabla I(x,y)\|$ 勾配強度画像

Shin Yoshizawa: shin@riken.jp

勾配強度画像生成ヒント

1. pgm画像を読み込む:
   1. SimpleImage.hをincludeし入力画像用にメモリ確保を行う:
      Image *A = new Image();
   2. pgmio.hをincludeしgetPGM(Image *,char *)を使う.
2. x,y方向の微分を差分近似し画像B,Cとする:
   1. 画像B,CをAと同じサイズで確保する:
      Image *B = new Image(A->sx,A->sy);
      Image *C = new Image(A->sx,A->sy);
   2. forの二重ループ(0<=i<A->sy, 0<=**j<A->sx-1**)でBの中身(輝度値)を作る: B->img[i][j]=A->img[i][j+1]-A->img[i][j];
   3. forの二重ループ(0<=**i<A->sy-1**, 0<=j<A->sx)でCの中身(輝度値)を作る: C->img[i][j]=A->img[i+1][j]-A->img[i][j];
3. 勾配ベクトルの大きさを画像Dとする:
   1. 画像DをAと同じサイズで確保する:
      Image *D = new Image(A->sx,A->sy);

Shin Yoshizawa: shin@riken.jp

勾配強度画像生成ヒント2

3. 勾配ベクトルの大きさを画像Dとする:
   2. forの二重ループ(0<=i<A->sy, 0<=j<A->sx)でDの中身(輝度値)を作る:

```
D->img[i][j]=sqrt(B->img[i][j]*B->img[i][j]+C->img[i][j]*C->img[i][j]);
```

3. 0-255にしてセーブはエンボス画像生成と同じ方法.
   1. 出力用のoutをAと同じサイズで確保する:
      Image *out = new Image(A->sx,A->sy);
   2. Dの輝度値の最小と最大を計算する:
   3. forの二重ループでoutの中身を作る:

```
out->img[i][j]=255.0*(D->img[i][j]-min)/fabs(max-min);
```

   4. savePGM(Image*, char *)を使ってセーブする.
   5. newしたクラスはdeleteする事: delete A; delete B; delete C; delete D; delete out;

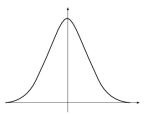
Shin Yoshizawa: shin@riken.jp

出来る人のための課題3

✓ Gaussianフィルタ↓の[プログラムを一から作成！]

連続式:

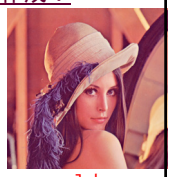
$$I^{\text{new}}(\mathbf{x}) = \frac{\int_{\Omega} g_\sigma(|\mathbf{x}-\mathbf{y}|) I(\mathbf{y}) d\mathbf{y}}{\int_{\Omega} g_\sigma(|\mathbf{x}-\mathbf{y}|) d\mathbf{y}},$$



ガウス関数

$$g(r) = \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}}$$

Smoothingパラメータ σ

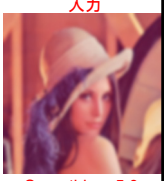


入力

離散化式:

$$I^{\text{new}}(i,j) = \frac{\sum_{y=-r}^{y=r} g(i-y)\left(\sum_{x=-r}^{x=r} g(j-x) I(i-x, j-y)\right)}{\sum_{y=-r}^{y=r} g(i-y)\left(\sum_{x=-r}^{x=r} g(j-x)\right)},$$

重み付平均の半径 r



Smoothing, 5.0

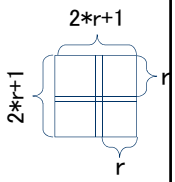
Shin Yoshizawa: shin@riken.jp

Gaussianフィルタヒント

✓ Gaussianフィルタ↓のプログラムを一から作成！

1. pgm画像を読み込む、画像Aとする.
2. ガウス関数画像Bを作る:

```
Image *B = new Image((2*r+1),(2*r+1));
double wsum=0.0;
for(i=-r;i<=r;i++)for(j=-r;j<=r;j++){
B->img[i+r][j+r] = exp(-(i*i+j*j)/(2*sigma*sigma));
  wsum+=B->img[i+r][j+r];
 }
```



3. BとAを畳み込む(重み付和を計算する):

```
for(i=0;i<A->sy;i++)for(j=0;j<A->sy;j++){out->img[i][j]=0.0;
    for(y=-r;y<=r;y++)for(x=-r;x<=r;x++)
        if((i+y)>=0&&(i+y)<A->sy&&(j+x)>=0&&(j+x)<A->sx)
            out->img[i][j]+=A->img[i+y][j+x]*B->img[y+r][x+r]/wsum;
 }
```

セーブやoutの確保等は勾配強度画像などと同じ、ただし0-255に変換ではなくカット.

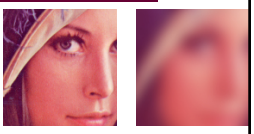
Shin Yoshizawa: shin@riken.jp

出来る人のための課題4

✓ Laplacianフィルタ↓のプログラムを一から作成！

連続式(拡散方程式):

$$\frac{\partial I(\mathbf{x},t)}{\partial t} = \Delta I(\mathbf{x},t),$$



離散式(拡散方程式の陽的前進一次差分近似):

$$I^{n+1}(i,j) = I^{n}(i,j) + \varepsilon(-9I(i,j) + \sum_{y=-1}^{y=1}\sum_{x=-1}^{x=1} I(i+y,j+x))/8,$$

ステップサイズパラメータ $\varepsilon < 0.5$

m回繰り返し適用する.

Shin Yoshizawa: shin@riken.jp

Lapalcianフィルタヒント

1. pgm画像を読み込む:画像Aへ.
2. n+1回目とn回目のテンポラリー用画像をC,Bとする:
   1. 画像B,CをAと同じサイズで確保する:
      Image *B = new Image(A->sx,A->sy);
      Image *C = new Image(A->sx,A->sy);
   2. forの二重ループ(0<=i<A->sy, 0<=j<A->sx)でBを初期化する: B->img[i][j]=A->img[i][j];
3. m回繰り返しフィルタを適用する.

```
for(n=0;n<m;n++){
   for(i=1;i<sy-1;i++)for(j=1;j<sx-1;j++){double sum=0.0;
for(y=-1;y<=1;y++)for(x=-1;x<=1;x++)sum += B->img[i+y][j+x];
C->img[i][j] = B->img[i][j]+eps*(-9.0*B->img[i][j]+sum);
  }
for(i=0;i<sy;i++)for(j=0;j<sx;j++)B->img[i][j]=C->img[i][j];
}
```

セーブやoutの確保等は勾配強度画像などと同じ、ただし0-255に変換ではなくカット.