

情報デザイン専攻

画像情報処理論及び演習I

-前期課題の復習-

レポート、成績、後期の予定・予習

第12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/briect/Yoshizawa/Lectures/index.html
www.riken.jp/briect/Yoshizawa/Lectures/Lec12.pdf

- ① 後期の予定.
- ② 補講・成績について.
- ③ 演習: レポート1~3の内容.

↑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!

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5点加点(レポート点約25点分・出席約2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. } 理論は後期にやります。後期のレポートで出します!
 - Laplacianフィルタ.

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。

入力 $I(x, y)$

勾配強度画像 $\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$

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: $I^{new}(x) = \frac{\int_{\Omega} g_{\sigma}(|x-y|)I(y)dy}{\int_{\Omega} g_{\sigma}(|x-y|)dy}$

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

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):

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

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.

ICG-ARTS 発表

Shin Yoshizawa: shin@riken.jp

後期の予定1

- ✓ 周波数分解・ファイルI/O
- ✓ フィルタ処理・エッジ強調
- ✓ 計算Photography
- ✓ Artistic Stylization
- ✓ 動画像処理
- ✓ 幾何・形状・パターン認識



Timeline of research projects:

- Late 1980s: Advances in media emulation
- 1990: Semi-automatic painting systems
- 1997: Fully automatic painting
- 1998: Video painting
- 2000: Perceptual UI & segmentation
- 2002: Automatic perceptual
- 2005: Space-time video
- 2006: Anisotropy / filters
- 2010: NIMN 2010 Grand challenges, User evaluation

ICG-ARTS 発表

Shin Yoshizawa: shin@riken.jp

後期の予定2



特にフィルタ処理とフィルタを用いたスタイル化.



© New Line Productions, Inc.

ICG-ARTS 発表

Shin Yoshizawa: shin@riken.jp

後期の予定3

- 「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」



ICG-ARTS 発表

Shin Yoshizawa: shin@riken.jp

後期の休講

10月29日(水)及び11月5日(水)は、
休講ですm(_ _)m
試験期間(予定)に補講を2限分を行います。

ICG-ARTS 発表

Shin Yoshizawa: shin@riken.jp

レポート1のヒント

ICG-ARTS 発表

Shin Yoshizawa: shin@riken.jp

演習:レポート1の内容

www.riken.jp/briect/Yoshizawa/Lectures/index.html

- ✓ Q1(簡単): ppm画像をpgm画像へ変換するプログラムを作成.
- ✓ Q2(簡単): pgm画像を与えられた閾値を用いて2値化するプログラムを作成.
- ✓ Q3: pgm画像をHue画像へ変換するプログラムを作成.
- ✓ Q4: pgm画像の画像全体の輝度値の最大値、最小値、平均値、及び中央値を計算するプログラムを作成.

Shin Yoshizawa: shin@riken.jp

Q1: カラーからグレースケールへの変換

www.riken.jp/briect/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


Q2: 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, atof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

Q2: 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.

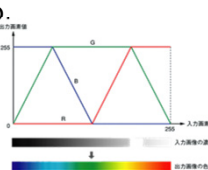


Shin Yoshizawa: shin@riken.jp

Q3: 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

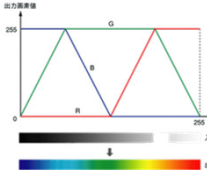
Q3: ヒント

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

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

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

- ✓ $y = ax + b$ 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.
- ✓ forの二重ループで変換し保存.



Shin Yoshizawa: shin@riken.jp

Q3: ヒント

- lena.pgmでそのままin->img[i][j]を変換したのが左の結果になります。

Shin Yoshizawa: shin@riken.jp

Q4: 統計

- 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。
- argvを使って、入力画像ファイル名を指定出来る事。
- ヒント: 中央値は、輝度値の値を大ききで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

Q4: ヒント

- 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
std::sort(val.begin(),mval.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

レポート2のヒント

Shin Yoshizawa: shin@riken.jp

演習: レポート2の内容

www.riken.jp/briect/Yoshizawa/Lectures/index.html

- ✓ Q1-10: 講義のスライドをよく読む or 自分で調べる:

参考書:

- 「デジタル画像処理」、CG-ARTS協会、2006.
- 「Digital Image Processing」, R. Gonzalez & R. Woods著, Pearson Edu. Inc., 2008.

- ✓ Q11: pgm画像を大津の方法を用いて二値化するプログラムを作成。
- ✓ Q12(最も簡単): ラベリングと細線化の実行結果を載せて考察。

Shin Yoshizawa: shin@riken.jp

Q11: 大津法のプログラムを作ってみよう!

www.riken.jp/briect/Yoshizawa/Lectures/index.html
www.riken.jp/briect/Yoshizawa/Lectures/Ex03.zip

- Ex03.zip内のOtsuBin.cxx: pgmファイルを大津の方法で二値化するプログラムの雛形ソースコード。
- [OtsuBin.cxx内のコメントを参考にプログラムを完成させる。](#)
- 「make」でコンパイル。
- lena.pgm、Cameraman.pgm、Kanji_Iri.pgmを大津の方法で二値化しよう!

大津法の閾値: 118, 99, 127でもOK.: 演習のヒントに従うとこっち!

大津法の閾値: 117 大津法の閾値: 88 大津法の閾値: 126

Shin Yoshizawa: shin@riken.jp

Q11:OtsuBin.cxx内の説明

```

int main(int argc,char **argv){
  /* 入力用の画像ファイル */
  Image *in = new Image(1);
  /* argvの数を数え、第一引数 argv[1]に入力ファイル名 */
  getPSM(in,argv[1]);
  /* 入力画像と同じサイズで出力用の画像クラス(オブジェクト)を作成 */
  Image *out = new Image(in->w,in->h);
  /* 大津法で閾値を計算 */
  int otsu_threshold = find_hist_threshold(in,255);
  printf("otsu = %d\n",otsu_threshold);
  /* 閾値を二値化 */
  int i;
  for(i=0;i<in->w;i++)
    for(j=0;j<in->h;j++){
      /* hist[i][j]での確率P->img[i][j]が閾値以上、未決で二値化 */
      if((img[i][j]>otsu_threshold) || !otsu_threshold){
        out->img[i][j] = 255.0;
      }else{
        out->img[i][j] = 0.0;
      }
    }
  /* 出力画像を第二引数argv[2]の名前でppm画像として保存 */
  savePPM(out,argv[2]);
  /* メモリの開放 */
  delete in;
  delete out;
  return 0;
}

```

makeHistogram()は画像in、頻度表の配列hist、及びピンの数Nを与えてhistの中へ結果を保存。

```

void makeHistogram(long *hist,int N,Image *in){
  int i,j;
  for(i=0;i<N;i++){
    for(j=0;j<in->h;j++){
      /* 確率を計算 */
      double val = (double)(in->img[i][j])/(double)(in->w*in->h);
      /* 数値の追加 */
      hist[i] += (int)(val);
    }
  }
  /* 閾値を計算 */
  double m1 = 0.0, m2 = 0.0;
  for(i=0;i<N;i++){
    m1 += i*hist[i];
    m2 += i*i*hist[i];
  }
  /* 閾値を計算 */
  double m1_bar = (double)m1/N;
  double m2_bar = (double)m2/N;
  double S = m2_bar - m1_bar*m1_bar;
  /* もしもS>SmaxならばSmax=S, Tmax=Tとする */
  if(S>Smax) Smax=S, Tmax=i;
  hist[val]++;
}

```

Shin Yoshizawa: shin@riken.jp

Q11:OtsuBin.cxx内の説明2

```

double getSmax(int thr,int N,long *hist){
  long m1=0, m2=0;
  for(i=0;i<N;i++){
    m1 += i*hist[i];
    m2 += i*i*hist[i];
  }
  double m1_bar = (double)m1/N;
  double m2_bar = (double)m2/N;
  double S = m2_bar - m1_bar*m1_bar;
  /* もしもS>SmaxならばSmax=S, Tmax=Tとする */
  if(S>Smax) Smax=S, Tmax=i;
  return Smax;
}

```

find_hist_threshold()で閾値が0~N-1までgetSmax()にてクラス間分散を計算

- 画像からヒストグラムを作成。ピンの数をNとする。
- 閾値が0のときのクラス間分散を計算しその値をSmax。そのときの閾値をTmaxとする。
- for(i=1;i<N;i++){
 1. 閾値がiのときのクラス間分散を計算しSとする。

$$S = \frac{m_1(m_1 - m_1^2) + m_2(m_2 - m_1^2)}{m_1 + m_2}$$

$$S = \frac{m_1^2(m_1 - m_1^2) + m_2^2(m_2 - m_1^2)}{(m_1 + m_2)^2}$$

$$S = \frac{m_1^2(m_1 - m_1^2) + m_2^2(m_2 - m_1^2)}{(m_1 + m_2)^2}$$
- もしもS>SmaxならばSmax=S, Tmax=Tとする。
- Tmaxが大津の閾値となる。

Shin Yoshizawa: shin@riken.jp

Q11:Ex03.zip内のOtsuBin.cxx中で、

getSmax()内の平均(m1,m2)を求める計算にて、ゼロでの割り算を防ぐため、mn1がゼロの場合はm1もゼロ、mn2がゼロの場合はm2もゼロで計算してください。

大津法の正解閾値はLec06.pdfの値より一つ値が大きいてもOKです。

大津法の閾値: 118, 99, 127でもOK: 演習のヒントに従うとこっち!

大津法の閾値: 117 大津法の閾値: 88 大津法の閾値: 126

Shin Yoshizawa: shin@riken.jp

Q12:プログラムの説明

Ex04.zipをダウンロード→解凍。
コンパイルは「make」、詳細はMakefileを見てください。

- LabelingRemoveSmall.cxx(引数の数3) pgmを大津法+ラベリング(8連結)+第三引数以下の領域サイズを一つにまとめる(小さい面積の領域を統合)+ラベル毎に疑似カラーでppmで保存。
 - 実行方法: ./LabelingRemoveSmall 入力pgm 出力ppm 削除する領域の面積閾値(int)
- Thinning.cxx(引数の数2): 大津法+Hilditchの細線化。
 - 実行方法: ./Thinning 入力pgm 出力pgm
- ヘッダーファイル: otsu.h: 大津法、label.h: ラベリング、color.h: 疑似カラー、thinning.h: 細線化。
 - 実装の細部に興味がある人は見てください。

Shin Yoshizawa: shin@riken.jp

Q12:プログラムの実行例

領域抽出+ラベリングを行うと、表示の綺麗さだけでなく定量的な解析が可能になる(数、面積、境界形状の長さや曲率など)。

領域数220 領域数19 領域数14 領域数9

閾値:0 閾値:30 閾値:60 閾値:120

領域数220

入力 大津法二値化 ラベリング(ID=輝度値) ラベリング疑似カラー

Shin Yoshizawa: shin@riken.jp

Q12:プログラムの実行例2

細線化は幅が1画素の線になる。線あり:黒・なし:白の表示。

入力 大津法二値化 細線化

Shin Yoshizawa: shin@riken.jp

レポート3のヒント Lec11.pdfと同じ

Shin Yoshizawa: shin@riken.jp

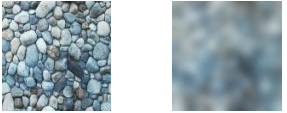

演習:レポート3の内容

www.riken.jp/briect/Yoshizawa/Lectures/index.html
Ex05及びEx06で端末にて「make clean」&「make」.

- ✓ Q4: Image Analogyを用いてオリジナルのエフェクト.
- ✓ Q1(簡単): 油絵効果、水彩画効果、線画効果、テクスチャー合成の4種類.
- ✓ Q2: Texture by Numbers.
- ✓ Q3: Poisson Image Editing.

Shin Yoshizawa: shin@riken.jp

Q4: Image Analogyでオリジナル効果

- Ex05内のSmoothingでtexture5.ppmをフィルタリング: 端末で
`./Smoothing texture5.ppm texture5_blur.ppm 5.0`

- ArtisticFilterを使って
`./ArtisticFilter texture5_blur.ppm texture5.ppm lena.ppm lena_r3q4.ppm 2.0 1000.0 2 display lena_r3q4.ppm`


Shin Yoshizawa: shin@riken.jp

Q4: Image Analogyでオリジナル効果2

- 例えば「Zebra効果」、もしも以下の様なzebra.ppmを持っていたら、
`./Smoothing zebra.ppm zebra_blur.ppm 5.0`

- ArtisticFilterを使って
`./ArtisticFilter zebra_blur.ppm zebra.ppm lena.ppm lena_r3q4.ppm 2.0 1000.0 2 display lena_r3q4.ppm`

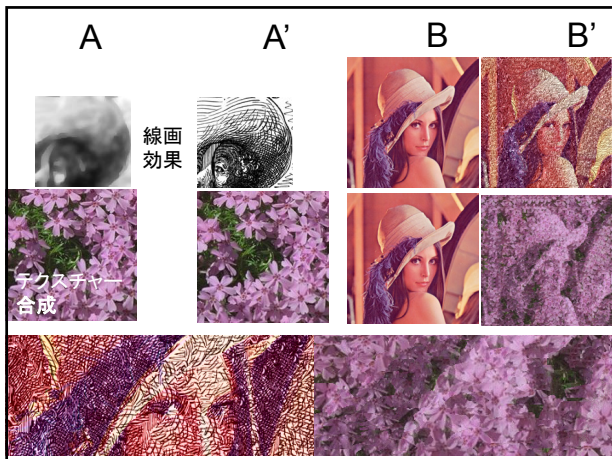

Shin Yoshizawa: shin@riken.jp

Q1: Image Analogy: Artistic Filters

- ✓ 端末でEx05に移動:もしもEx05をデスクトップで立ち上げていたら「cd ~/Desktop/Ex05」又はファイルブラウザのパスをコピーして端末に張り付けて「cd パス」でエンターキーを押す.

- 油絵効果**をlena.ppmに適用してみる: 端末で、
`./ArtisticFilter rhone-src.ppm rhone.ppm lena.ppm test1.ppm 2.0 1000.0 2`
 を打ち込んでエンターキーを押す. 実行が終了したら、端末で、
`display test1.ppm &`
 同様に、
- 水彩画効果**をlena.ppmに適用してみる:
`./ArtisticFilter watercolor-src.ppm watercolor.ppm lena.ppm test2.ppm 2.0 1000.0 2`
- 線画効果**をlena.ppmに適用してみる:
`./ArtisticFilter squire-blur.ppm squire.ppm lena.ppm test3.ppm 2.0 1000.0 2`
- テクスチャー合成**をlena.ppmに適用してみる:
`./TextureTransfer texture1.ppm texture1.ppm lena.ppm test3.ppm 2.0 1000.0 4 0.75`

	A	A'	B	B'
油絵効果				
水彩画効果				



Shin Yoshizawa: shin@riken.jp

Q2: NumberEditor & TextureByNumbers

- Image Analogy用TextureByNumbersのお絵かきGUI (Java).
- Ex06/NumberEditor/

- sh Run_NumberEditor.shで立ち上げてください。
- 画像を読み込む: File->Load ppm Image. [Ex05/darkclouds.ppm](#)を開いてみてください。
- お絵かき: 左ドラッグ: **木、岩、草原、空を違う色で塗ってみてください。**
 - 色を変える: 右下のSelectボタン。
 - ブラシのサイズを変える: 右のスクロールバー or マウスホイール。
 - 表示の透明度を変える: 下のスクロールバー。
- セーブ:File->Save Number Image. [A.ppm](#)という名前前で保存してください。

Shin Yoshizawa: shin@riken.jp

NumberEditor & TextureByNumbers2

- 追加のお絵かき: **木、岩、草原、空で使った色とほぼ同じ色で書き足してみてください。**
- マスク画像(ppm)をセーブ:File->Save Number Image. [B.ppm](#)という名前前で保存してください。
- A.ppmとB.ppmをEx05の下に移動(コピーでもカット&ペーストでもOK)してください。
- 端末を新たに立ち上げて、Ex05にcdで移動してください。もしもEx05をデスクトップで立ち上げていたら「cd ~/Desktop/Ex05」又はファイルブラウザのパスをコピーして端末に張り付けて「cd パス」でエンターキーを押す。

Shin Yoshizawa: shin@riken.jp

NumberEditor & TextureByNumbers3

- emacsでRun_TextureByNumbers.shを立ち上げて、以下の様書き換えてください。 ./TextureByNumbersの後の
 - 第一引数oxbow-mask.ppm は A.ppm
 - 第二引数oxbow.ppm は darkclouds.ppm
 - 第三引数oxbow-newmask.ppm は B.ppm
 - その後のppmファイル名も上のルールで変更してください。
- Run_TextureByNumbers.shをセーブ(上書き保存)してください。
- 端末にて「sh Run_TextureByNumbers.sh」で実行してみてください。

こんな感じなのが
出れば正解⇒

Shin Yoshizawa: shin@riken.jp

Q3: Poisson Image editing & MaskEditor

- PIE用マスク作成GUI (Java): Ex06/MaskEditor/

- sh Run_MaskEditor.shでMaskEditorを立ち上げてください。
- Source画像を読み込む: File->Load Sourceで[Ex06/images/Keira02.ppm](#)を開いてください。
- Target画像を読み込む:File->Load Targetで[Ex06/images/MonaLisa.ppm](#)を開いてください。
- 左クリックでPolylineを生成して[Keiraの顔領域を作成](#)してみましょう!

Shin Yoshizawa: shin@riken.jp

MaskEditor2

- PIE用マスク作成GUI (Java): Ex06/MaskEditor/

- Source画像の大きさと位置を合わせる: [Keiraの顔とMonaLisaの顔の大きさと位置を合わせてみよう!](#)
 - 右クリックでMove Picを選べば平行移動可能。
 - 右クリックでAddを選べばPolyline作成モードに戻れる。
 - マウスの真ん中ホールで拡大縮小。
 - Polylineの頂点は左クリックで移動可能。
 - 下のスクロールバーで表示の透明度を変更可能。

Shin Yoshizawa: shin@riken.jp

MaskEditor3

- マスク画像(ppm)とTargetと同じ大きさのSource画像(ppm)の二つの画像をセーブ: File->Save Masks: ソースとマスクをKeiraMonaという名前で作ってみよう!

注: セーブするファイル名に拡張子はいらない: ファイル名.ppmとファイル

- 端末でPoissonImageEditorを以下の様に動かして合成してみよう!
 - 端末を立ち上げてEx06へ移動: 「cd ~/Desktop/Ex06」.
 - ./PoissonImageEditor ./MaskEditor/KeiraMona.ppm ./MaskEditor/KeiraMona.ppm ./images/MonaLisa.ppm KM_PIE.ppm 1.0 0.0
 - display KM_PIE.ppm &

Source Mask Target 合成結果

Shin Yoshizawa: shin@riken.jp

補足資料(Lec08.pdf): ANNのコンパイル

www.riken.jp/brict/Yoshizawa/Lectures/Ex05.zip

まずはじめに、ANNをコンパイルする。

- Ex05.zipを展開する。
- Ex05内にann_1.1.2.zipがあるのでEx05内で展開する。
- 端末でEx05/ann_1.1.2に入る、もしもデスクトップに展開していたら、「cd ~/Desktop/Ex05/ann_1.1.2」.
- コンフィギュレーションを行う4.の後に端末で「sh Make-config」でエンターキー。
- コンパイルする5.の後に端末で「make linux-g++」と打ち込みエンターキーを押す.Ex05/ann_1.1.2/libの下にlibANN.aが出来れば成功。
- Ex05で端末にて「make clean」&「make」.

Shin Yoshizawa: shin@riken.jp

補足資料: Lec08.pdf

- ✓ Smoothing.cxx: ガウス平滑化を実行するプログラム: 引数3:
 - Smoothing 入力.ppm 出力.ppm 平滑化度合(double)
 - 平滑化度合のパラメータは0より大きな実数2.0~20.0ぐらいが実用的。
- ✓ EdgePreservingFilter.cxx: エッジ保存平滑化を実行: 引数3
 - EdgePreservingFilter 入力.ppm 出力.ppm エッジの大きさ(double)
 - エッジの大きさパラメータは0より大きな実数0.5~2.0ぐらいが実用的。

EdgePreservingFilter, 1.0 入力 Smoothing, 5.0

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:

$$B \rightarrow \text{img}[i][j] = 255.0 - A \rightarrow \text{img}[i][j];$$
3. Bを平行移動しAと合成する:

$$C \rightarrow \text{img}[i][j] = B \rightarrow \text{img}[i+t][j+t] + A \rightarrow \text{img}[i][j] - 128.0;$$
4. 0~255に正規化しpgmでセーブ:

$$\text{out} \rightarrow \text{img}[i][j] = 255.0 * (C \rightarrow \text{img}[i][j] - \min(C)) / \text{fabs}(\max(C) - \min(C));$$

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:

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

Shin Yoshizawa: shin@riken.jp

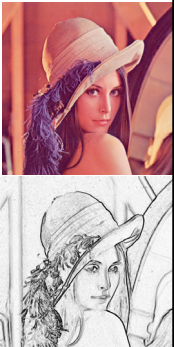
出来る人のための課題2

✓ 勾配強度画像生成↓のプログラムを一から作成!

- pgm画像を読み込む、画像Aとする.
- 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];
- 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] + C->img[i][j]*C->img[i][j]);
- 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));

$I(x, y)$ 入力

$\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$ 勾配強度画像



Shin Yoshizawa: shin@riken.jp

勾配強度画像生成ヒント

- pgm画像を読み込む:
 - SimpleImage.hをincludeし入力画像用にメモリ確保を行う:
Image *A = new Image();
 - pgmio.hをincludeしgetPGM(Image *, char *)を使う.
- x,y方向の微分を差分近似し画像B,Cとする:
 - 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);
 - 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];
 - 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];
- 勾配ベクトルの大きさを画像Dとする:
 - 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

- 勾配ベクトルの大きさを画像Dとする:
- 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]);
- 0-255にしてセーブはエンボス画像生成と同じ方法.
 - 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);
 - Dの輝度値の最小と最大を計算する:
 - 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);
 - savePGM(Image*, char *)を使ってセーブする.
 - 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!

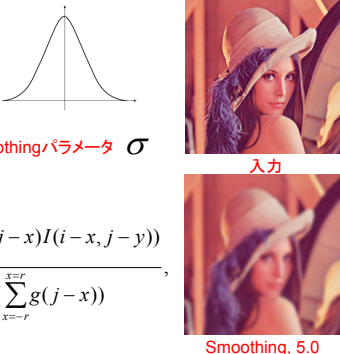
連続式: $I^{new}(x) = \frac{\int_{\Omega} g_{\sigma}(|x-y|)I(y)dy}{\int_{\Omega} g_{\sigma}(|x-y|)dy}$

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

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



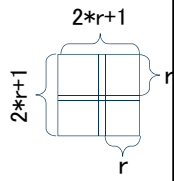
Shin Yoshizawa: shin@riken.jp

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!

- pgm画像を読み込む、画像Aとする.
- 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];
}
- 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.



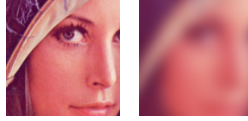
出来る人のための課題4



✓ 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.

©CG-ARTS 発表

Laplacian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.