

情報デザイン専攻

画像情報処理論及び演習I

-補習・復習-
レポート第4回+Linux コマンド

第13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/Lec10.pdf

- ① みんな来るまで...レポート第1~4回の質問.
- ② 前回までのみんなの成績.
- ③ Linuxコマンドの復習.
- ④ 前回の演習の続き: 画像合成 (Image Analogy+Poisson Image Editing).
[レポート1~3採点結果を取りに来てください.](#)

今日は第4回レポートの×切なので
みなさん頑張って出してくださいねーp(^)q

Shin Yoshizawa: shin@riken.jp

第13~15回の予定

✓ 要望があったので、レポート内容の
補習・復習に変更します。

©S. Yoshizawa, RIKEN ©IIPImage

予定していた「圧縮・周波数分解・符号化・ファイル/O」は後期にやります。

Shin Yoshizawa: shin@riken.jp

何でも質問してねー

レポート第1~4回の質問を受け付けます!
レポート1~3採点結果(まだの人は)取りに来てください。
9:30頃まで...

✓ 出来る人・わかる人は第4回のレポートを進めてください。
✓ 第4回レポートを既に提出した人はエンボス画像生成!
のプログラムを一から作ってみましょう!

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

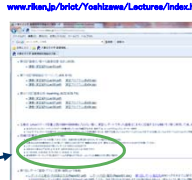
成績の計算方法について

出席4割、レポート6割。

成績点数(基本は...もしかしたらゲタが...)=
 出席日数 × (20/7) + レポート4回分の合計点 × (3/20)

- ✓ 出席点は40点を休講を抜かした14で割って一回あたり(20/7)点です。補講日も加算します。遅刻した日は(20/7)の代わりに0.8 × (20/7)で計算してください。
- ✓ レポートの点は60点を400点で割ってレポートの1点あたり(3/20)点です。
- ✓ 合計点の小数点以下は切り上げします。合計点100点以上の方は100点です(^)
- ✓ S: 100-90点, A: 89-80点, B: 79-70点, C: 69-60点, D: 59-0点。
- ✓ 単位取得ボーダーの人や「あと数点で一つ上の評価なので何とか...」という人は補講日(7/29:5限)に相談可。

↑は授業のHPIにもあります



Shin Yoshizawa: shin@riken.jp

前回までのみんなの成績

前回まで出席12回+レポート3回の点数は51名中...

- ✓ 既に単位取得確定(60点以上)の人: 28名。
 - 既にA以上確定の人: 数名。
 - 既にB以上確定の人: 10名程。
- ✓ あと少して単位取得確定の人: 11名。
 - 残りの授業出席&レポート第4回提出で単位取得は大丈夫。
- ✓ 頑張れば単位OKな人: 6名。
 - レポート1~3のQ9 & Q10再提出すれば単位はOK。
- ✓ (ほぼ)不合格確定の人: 6名。
 - レポート今から全部提出&高得点でないダメ。

その他履修中止: 17名! (P.D. q)

Shin Yoshizawa: shin@riken.jp

復習: Linuxコマンド

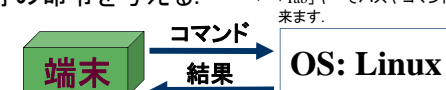

✓ この内容は「演習補足資料(ex01_2.pdf)」でやった内容の復習です。
10:00頃まで

- ✓ 出来る人・わかる人は第4回のレポート or エンボス画像生成プログラム作成を進めてください。
- ✓ フォルダ・ディレクトリ構造/パスと端末:
 - `cd` : フォルダ・ディレクトリの移動。
 - `ls` : フォルダ・ディレクトリの中身を見る。
 - `pwd` : 現在のフォルダ・ディレクトリのパスを見る。

Shin Yoshizawa: shin@riken.jp

コンソール(端末)とエディター(emacs)

- ✓ コンソール(端末): `cd`, `ls`, `pwd`等のLinuxコマンドを入力し「Enter」キーを押す事でOS(オペレーティングシステム)にファイル操作やプログラムのコンパイル等の命令を与える。
 - ✓ 「Tab」キーでパスやコマンドを補間出来ます。
- ✓ エディター(emacs): プログラムや文章を書くツール。テキストでプログラムを入力→ファイルとして保存。

Shin Yoshizawa: shin@riken.jp

Linuxでのプログラミングの流れ: 1

- ① コンソール(端末)を立ち上げる: 画面左下のコンソールアイコンをクリック。
- ② 端末: でemacsを立ち上げる: 端末に「emacs &」と打ち込み「Enter」キーを押す。
- ③ emacsでプログラム(ソースファイル)を書く。
- ④ emacsからソースファイルを保存する。
- ③ プログラムの作成・編集

ソースファイル: HelloWorld.cxx

④ ソースファイルの保存

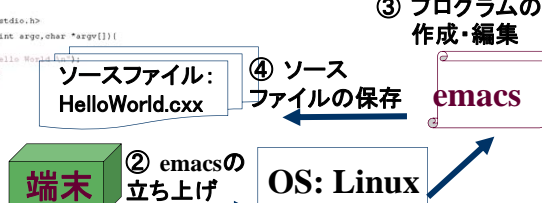
emacs

② emacsの立ち上げ

OS: Linux

```

#include<stdio.h>
int main(int argc, char *argv[])
{
    printf("Hello World\n");
    return 0;
}
  
```



Shin Yoshizawa: shin@riken.jp

Linuxでのプログラミングの流れ: 2

- ⑤ 端末でソースファイルをコンパイルして実行ファイルを作る: 「g++ ソースファイル名」 or 「g++ -o 実行ファイル名 ソースファイル名」. 実行ファイル名を指定しない場合は「a.out」という名前の実行ファイルが作成される。
- ⑥ 端末で実行ファイルを実行する: 「./a.out」 or 「./実行ファイル名」

実行ファイル: a.out

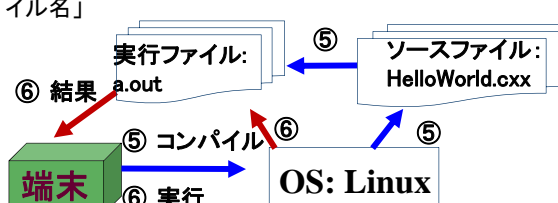
⑤ 結果

⑤ コンパイル

⑥ 実行

OS: Linux

ソースファイル: HelloWorld.cxx



Shin Yoshizawa: shin@riken.jp

ディレクトリー構造

ルートディレクトリー: /

各ユーザーのホームディレクトリー: /home/ユーザー名

home

ユーザー名1

Desktop

学籍番号_Ex01

HelloWorld.cxx

ココ!

実行!

- ✓ コンソールは立ち上げたら(自分の)ユーザーのホームディレクトリーにいます: pwdの結果は「/home/ユーザー名1」.
- ✓ 端末から動かしたプログラム(emacs)やコマンドはコンソールがいるディレクトリーで動作します.

ディレクトリー: Windowsのフォルダーと同じ.

エディター(emacs)

ファイル

コンソール(端末)

Shin Yoshizawa: shin@riken.jp

ディレクトリー構造

ルートディレクトリー: /

各ユーザーのホームディレクトリー: /home/ユーザー名

home

ユーザー名1

Desktop

学籍番号_Ex01

HelloWorld.cxx

移動!

- ✓ cdで今のコンソールの位置を移動します.
- ✓ 最初に立ち上げたemacsの位置はそのままです.
- ✓ 「cd ディレクトリー名」で移動します.
- ✓ : pwdの結果は「/home/ユーザー名1/学籍番号_Ex01」

ディレクトリー: Windowsのフォルダーと同じ.

エディター(emacs)

ファイル

コンソール(端末)

Shin Yoshizawa: shin@riken.jp

ディレクトリー構造:絶対パス・相対パス

ルートディレクトリー: /

各ユーザーのホームディレクトリー: /home/ユーザー名

home

ユーザー名1

ユーザー名2

Desktop

学籍番号_Ex01

HelloWorld.cxx

- ✓ **絶対パス:** ルートディレクトリーから全てのディレクトリーを含んだパス.
- ✓ **相対パス:** 端末の自分の位置からの相対的パス.
- ✓ 「./」は今の、「../」は一つ上. 「cd /home/ユーザー名1/学籍番号_Ex01」 「emacs /home/ユーザー名1/学籍番号_Ex01>HelloWorld.cxx」
- ✓ **相対パス**は端末やemacs(動かしているプログラムの)今の位置からパスを指定します.
端末1:「cd ../ユーザー名2」、端末2:「cd ../学籍番号_Ex01」、端末3「cd ../ユーザー名1」、
端末1:「emacs ../HelloWorld.cxx」、端末2:「emacs ../学籍番号_Ex01>HelloWorld.cxx」.
- ✓ cd, g++, emacsの後に**絶対パス**を入れる事(引数として実行)で端末の**今の位置(pwd)**とは**関係なし**でコマンドを実行したりファイルを開けたりします.

Shin Yoshizawa: shin@riken.jp

Linuxコマンドレッスン

- ✓ この内容は「[演習補足資料\(ex01_2.pdf\)](#)」でやった内容の復習です.
- ✓ 私がセンターモニターで打ち込むコマンドを真似してみましょう!
- ✓ 出来る人・わかる人は第4回のレポートを進めてください.
- ✓ フォルダー・ディレクトリー構造/パスと端末:
 - cd :フォルダー・ディレクトリーの移動.
 - ls :フォルダー・ディレクトリーの中身を見る.
 - pwd :現在のフォルダー・ディレクトリーのパスを見る.

Shin Yoshizawa: shin@riken.jp

演習:前回の続き

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec10.pdf
 Image Analogy+Poisson Image Editing.

Image Analogy:
www.riken.jp/brict/Yoshizawa/Lectures/Lec07.pdf
www.riken.jp/brict/Yoshizawa/Lectures/Ex05.zip

Poisson Image Editing+(NumberEditor):
www.riken.jp/brict/Yoshizawa/Lectures/Lec08.pdf
www.riken.jp/brict/Yoshizawa/Lectures/Ex06.zip

Shin Yoshizawa: shin@riken.jp

Image Analogy: Artistic Filters



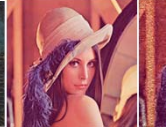
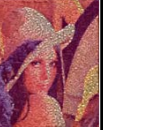



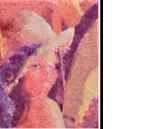

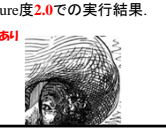
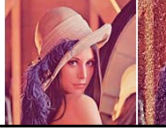

Shin Yoshizawa: shin@riken.jp

演習: 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。

1. **油絵フィルタ**をlena.ppmに適用してみる: 端末で、**訂正あり**
`./ArtisticFilter rhone-src.ppm rhone.ppm lena.ppm test1.ppm 1.0 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、

2. **水彩画フィルタ**をlena.ppmに適用してみる:
`./ArtisticFilter watercolor-src.ppm watercolor.ppm lena.ppm test2.ppm 1.0 1000.0 2`
3. **線画フィルタ**をlena.ppmに適用してみる:
`./ArtisticFilter squire-blur.ppm squire.ppm lena.ppm test3.ppm 1.0 1000.0 2`

| A | A' | B | B' |
|---|--|---|---|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

注意: この例は全てTexture度 2.0での実行結果。
訂正あり

Shin Yoshizawa: shin@riken.jp

Poisson Image Editing

Shin Yoshizawa: shin@riken.jp

重要: 演習: MaskEditor & NumberEditor

1. 端末にて「tcsh」と打ち込んでエンターキー。
2. 端末にて「setenv LANG C」と打ち込んでエンターキー。
3. 「sh Run_MaskEditor.sh」 or 「sh Run_NumberEditor.sh」

Shin Yoshizawa: shin@riken.jp

演習: MaskEditor

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

1. sh Run_MaskEditor.shでMaskEditorを立ち上げてください。
2. Source画像を読み込む: File->Load SourceでEx06/images/Keira02.ppmを開いてください。
3. Target画像を読み込む: File->Load TargetでEx06/images/MonaLisa.ppmを開いてください。
4. 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

演習: MaskEditor

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

1. 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!
 1. 右クリックでMove Picを選べば平行移動可能。
 2. 右クリックでAddを選べばPolyline作成モードに戻れる。
 3. マウスの真ん中ホールで拡大縮小。
 4. Polylineの頂点は左クリックで移動可能。
 5. 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

演習: MaskEditor

- 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 KE_PIE.ppm &

Source Mask Target 合成結果

Shin Yoshizawa: shin@riken.jp

演習: Image Analogy + PIEでリアルな合成

EdgePreservingFilterとArtisticFilterを使ってよりリアルな合成をしてみよう!

- EdgePreservingFilterでMonaLisa.ppmをフィルタリング: 端末で
~/Desktop/Ex05/EdgePreservingFilter ./images/MonaLisa.ppm ML_EP.ppm 1.0
- display ML_EP.ppm &

Shin Yoshizawa: shin@riken.jp

演習: Image Analogy + PIEでリアルな合成

EdgePreservingFilterとArtisticFilterを使ってよりリアルな合成をしてみよう!

- MonaLisa.ppmの代わりにML_EP.ppmで合成(ソース、マスクはそのまま):
./PoissonImageEditor ./MaskEditor/KeiraMona.ppm ./MaskEditor/KeiraMona.ppm ML_EP.ppm KM_PIE_EP.ppm 1.0 0.0
- display KM_PIE_EP.ppm &

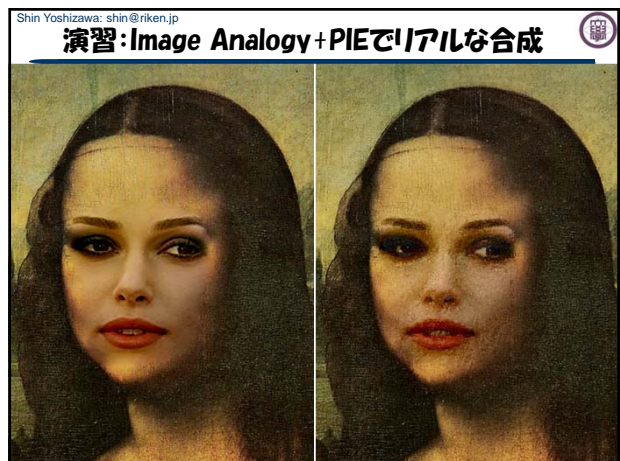
Source Mask Target 合成結果

Shin Yoshizawa: shin@riken.jp

演習: Image Analogy + PIEでリアルな合成

- Artistic Filterで細部を復元: A: ML_EP.ppm A': ./images/MonaLisa.ppm B: KM_PIE_EP.ppm
~/Desktop/Ex05/ArtisticFilter ML_EP.ppm ./images/MonaLisa.ppm KM_PIE_EP.ppm result.ppm 2.0 1000.0 2 **訂正あり**
- display result.ppm &

A A' B B'



Shin Yoshizawa: shin@riken.jp

復習: Image Analogy

✓ 様々なフィルタ処理が可能!

GA, Hartmann et al., SIGGRAPH 2001.

Shin Yoshizawa: shin@riken.jp

Image Analogy: Texture by Numbers

Shin Yoshizawa: shin@riken.jp

演習: 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

演習: NumberEditor & TextureByNumbers

- 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

演習: NumberEditor & TextureByNumbers

- emacsでRun_TextureByNumbers.shを立ち上げて、以下の様に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 **訂正あり**
 - 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.

Shin Yoshizawa: shin@riken.jp

来週の予定

✓ Cプログラムの引数とppm画像.
✓ レポート1~3のQ9とQ10について.

✓ ↑が出来る人・わかる人の演習課題も、ちゃんと用意します.

- 簡単な画像処理プログラミング:
 - エンボス画像生成、Gradient画像生成、Laplacian・Gaussianフィルタなど.