情報デザイン専攻

画像情報処理論及び演習I

-画像合成・類推-
Blending/Poisson Image Editing

第10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

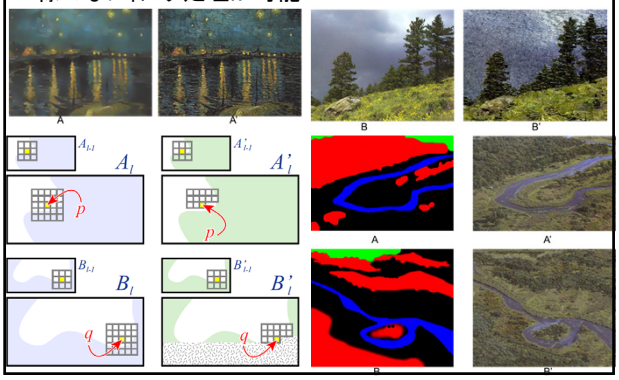
今日の授業内容

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec10.pdf

① Poisson Image Editing+演習.

前回の復習：Image Analogy

✓ 様々なフィルタ処理が可能！



前々回の復習：単純な合成

✓ 色の平均：$I^{new}(\mathbf{x}) = (I_1(\mathbf{x}) + I_2(\mathbf{x}))/2$

✓ Alpha-Blending：透明度を画素の位置により線形補間.

前々回の復習：クロスディゾルブ(Cross-Dissolve)

✓ 複数画像に対する変形結果のディゾルブを計算する事.

今回はBlendingについて

Shin Yoshizawa: shin@riken.jp

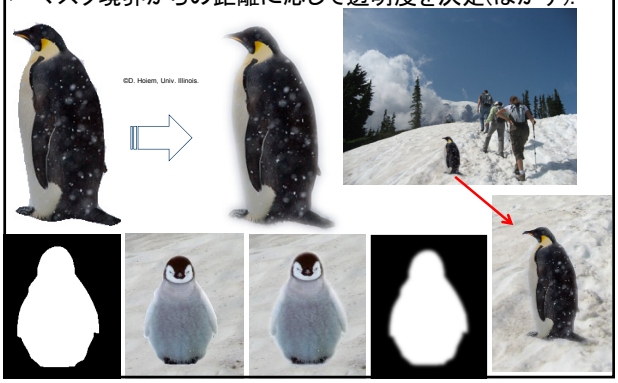
領域抽出だけで頑張るのは難しい…

✓ 髪の毛や繊維質、森の木々等の複雑な境界を自動的 or マニュアルで抽出しマスクを作成するのは困難.

Shin Yoshizawa: shin@riken.jp

Feathering

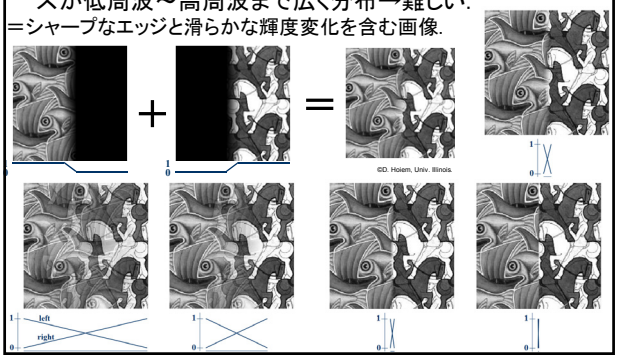
✓ マスク境界からの距離に応じて透明度を決定(ぼかす).



Shin Yoshizawa: shin@riken.jp

Blendingの方法がKey!

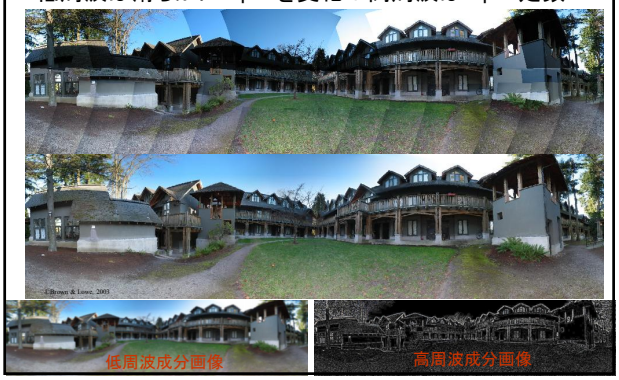
✓ 最適な幅や量は画像中の特徴サイズに依存=特徴サイズが低周波～高周波まで広く分布→難しい.
=シャープなエッジと滑らかな輝度変化を含む画像.



Shin Yoshizawa: shin@riken.jp

2-bands Blending

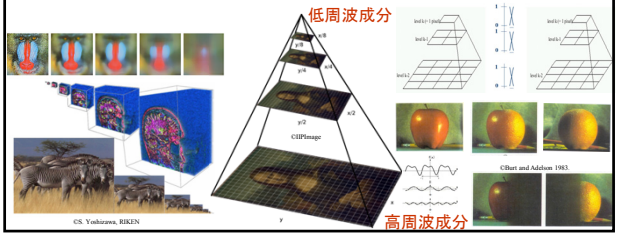
✓ 低周波は滑らかにAlphaを変化＋高周波はAlpha定数.



Shin Yoshizawa: shin@riken.jp

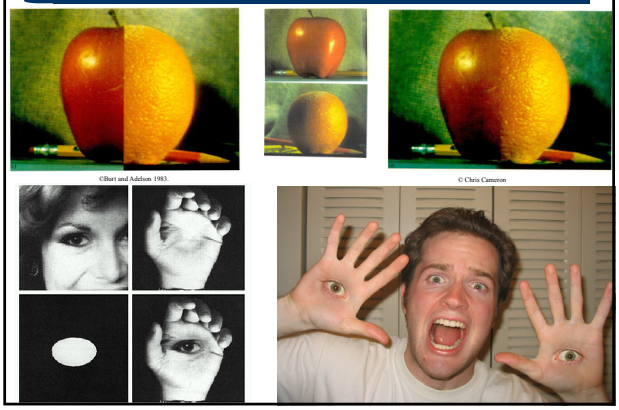
Pyramid Blending

- ✓ Multi-bands Blending: 周波数毎に異なる幅でBlending.
- ✓ **Pyramid Blending**: ピラミッドの各階層(周波数)で低周波はゆっくり(長い幅)、高周波は速く(短い幅)Blendさせる.
  - Gaussian/Laplacian Pyramid: 画像を平滑化して半分のサイズに再サンプルリングする操作を繰り返して作成: 後期の周波数分解の授業でもう少し詳しくやります.



Shin Yoshizawa: shin@riken.jp

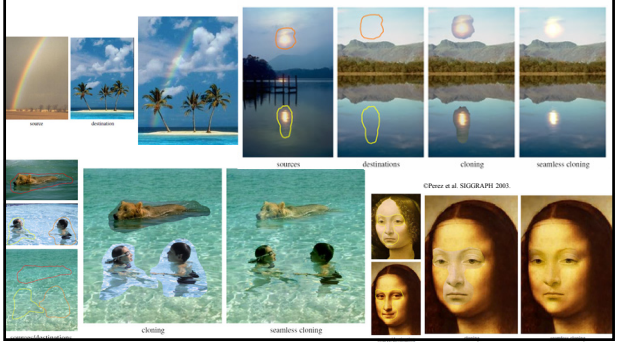
Pyramid Blending



Shin Yoshizawa: shin@riken.jp

重要:Poisson Image Editing

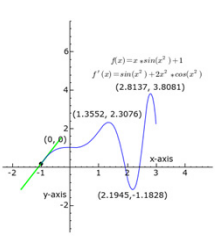
- ✓ Idea:良いBlendingはSource画像の勾配(Gradient=エッジ)を可能な限り保持する事が重要.

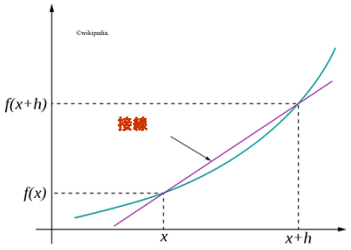


Shin Yoshizawa: shin@riken.jp

復習:微分

- ✓ 微分は与えられた関数の変数に対する変化を計算.
  - 1階微分の定義: $\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$





  - 1階微分は速度・傾き・接線を表す.

Shin Yoshizawa: shin@riken.jp

復習:微分2

- ✓ 例えば多項式なら…

$$f(x) = ax^3 + bx^2 + c \Rightarrow \frac{\partial f(x)}{\partial x} = 3ax^2 + 2bx$$

$$f(x) = a_1 x^n + a_2 x^{n-1} + a_3 x^{n-2} + \ldots + a_{n-2} x^2 + a_{n-1} x + a_n \Rightarrow$$

$$\frac{\partial f(x)}{\partial x} = n a_1 x^{n-1} + (n-1) a_2 x^{n-2} + (n-2) a_3 x^{n-3} + \ldots 2 a_{n-2} x + a_{n-1}$$

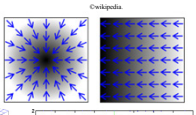
- ✓ 複数回の微分もある:
  - 2回微分: 加速度・傾きの傾き・曲がり具合.

$$\frac{\partial^2 f(x)}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f(x)}{\partial x}\right) = \frac{\partial}{\partial x}\left(\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}\right)$$

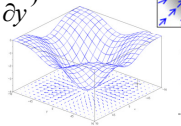
Shin Yoshizawa: shin@riken.jp

重要:勾配:Gradient

- ✓ 勾配(Gradient): スカラー場の各点で変化が最大の方向と変化率を大きさに持つベクトル場.



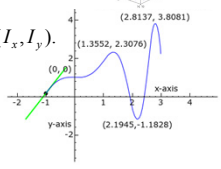
- ✓ 勾配作用素: $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$

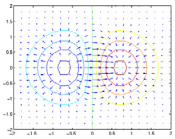


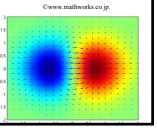
- ✓ 勾配ベクトルの表記:

$$\nabla I = \nabla I(x,y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$$

$$= \left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y}\right) = (I_x, I_y).$$







- ✓ 勾配の大きさ:

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

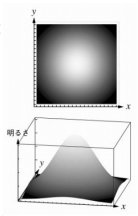
Shin Yoshizawa: shin@riken.jp

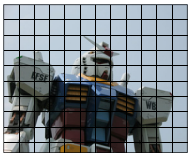
復習：デジタル画像の数式表現

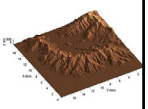
輝度値の配列表現：

int $I[sy][sx]$;

double $I[sy][sx]$;







輝度値の数式表現：高さ関数

$z = I(x, y)$ **又は** $z = I(\mathbf{x}), \quad \mathbf{x} = (x, y)$

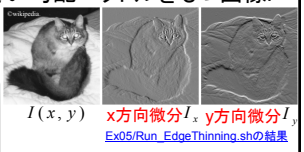
カラー画像： $\mathbf{z} = \mathbf{I}(x, y) = (R(x, y), G(x, y), B(x, y))$

又は $\mathbf{z} = \mathbf{I}(\mathbf{x}) = (R(\mathbf{x}), G(\mathbf{x}), B(\mathbf{x})), \quad \mathbf{x} = (x, y)$

Shin Yoshizawa: shin@riken.jp

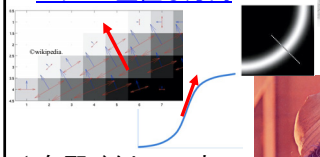
重要：画像の勾配：Gradient Image

- ✓ 画像の勾配: 画像を高さ関数と考えたときの勾配ベクトル場、画像のエッジ部分で大きい勾配ベクトルをもつ画像.
  後期のエッジ抽出でもう少し詳しくやります.
- ✓ 勾配ベクトルの方向: 画像エッジと垂直な方向.



$I(x, y)$ x方向微分I_x y方向微分I_y

Ex05/Run_EdgeThinning.shの結果



- ✓ 勾配ベクトルの大きさ=エッジ強度:

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

入力 $I(x, y)$

エッジ強度画像 $\|\nabla I(x, y)\|$

Shin Yoshizawa: shin@riken.jp

Laplace方程式・Poisson方程式

- ✓ ラプラス作用素(Laplacian): 滑らかさを記述.

$$\Delta = \nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \qquad \Delta I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

- ✓ 発散、湧き出し(Divergence): 注:ベクトルに対する作用素.

$$\mathrm{div} = \nabla \cdot = \frac{\partial}{\partial x} + \frac{\partial}{\partial y} \qquad \mathrm{div}\ \mathbf{v} = \nabla \cdot \mathbf{v} = \frac{\partial p}{\partial x} + \frac{\partial q}{\partial y}, \quad \mathbf{v}(x, y) = (p(x, y), q(x, y))$$

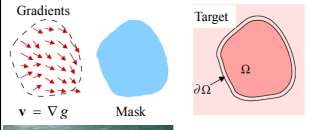
- ✓ Laplace方程式：自然科学の多くの分野で重要. $\Delta I = 0$
- ✓ Poisson方程式: Laplace方程式の右辺が関数. $\Delta I = g$
- ✓ 解くには境界条件（境界での値や微分値）が必要.
- ✓ **重要**：DivergenceのGradientはLaplacian:

$$\mathrm{div}\ \nabla = \Delta \qquad \Delta I = \mathrm{div}\ \nabla I$$

Shin Yoshizawa: shin@riken.jp

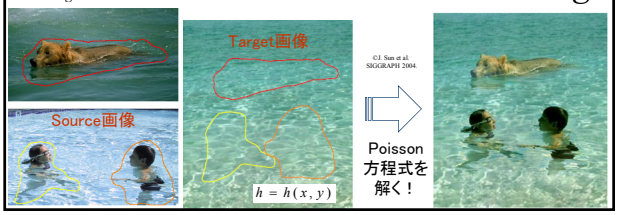
重要：Poisson Image Editingの原理

- ✓ Idea:良いBlendingはSource画像の勾配(Gradient=エッジ)を可能な限り保持する事が重要.



Source画像のGradient（マスク内）をTargetにコピーしマスク内だけTargetの境界条件で新しい輝度値I を解く.

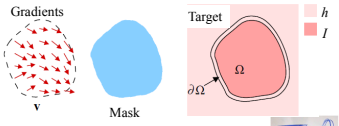
$$\Delta I = \mathrm{div}\ \nabla g$$



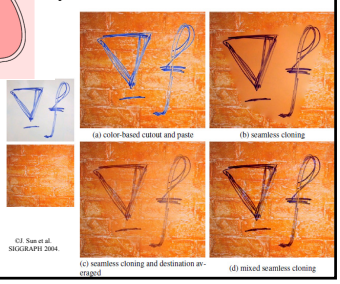
Shin Yoshizawa: shin@riken.jp

Mixing Gradients

- ✓ Target画像のテクスチャーを混ぜたい場合はSourceとTargetでGradientの強度が大きな方をPoisson方程式の右辺に使う.



$$\Delta I = \begin{cases} \mathrm{div}\ \nabla g & \text{if } \|\nabla g\| \geq \|\nabla h\| \\ \mathrm{div}\ \nabla h & \text{else} \end{cases}$$



- ✓ 顔の合成などTargetに特徴的な形状がある場合はダメ！

Shin Yoshizawa: shin@riken.jp

Poisson方程式の差分近似

- ✓ 微分方程式(Poisson方程式など)の数値解法で一番用いられているのは差分法(Finite Difference Scheme):
  - 差分近似：テイラー展開の高次の項を打ち切る→1次近似
    (Eulerの前進一次差分)は微分の定義からも得られる:

$$\frac{dI}{dx} = \lim_{h \to 0} \frac{I(x + h) - I(x)}{h} \approx \frac{I(x + h) - I(x)}{h}$$

  - 両辺で微分の差分近似を行い多元連立方程式を解く、特にPoisson Image Editingの場合は線形連立方程式になり疎な逆行列計算になる.

$$\Delta I = I_{xx} + I_{yy} \approx I(x+1, y) + I(x-1, y) + I(x, y+1) + I(x, y-1) - 4I(x, y)$$

$$\mathrm{div}\ \nabla g = \Delta g \approx g(x+1, y) + g(x-1, y) + g(x, y+1) + g(x, y-1) - 4g(x, y)$$

$$\Delta I = \mathrm{div}\ \nabla g \Rightarrow \mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

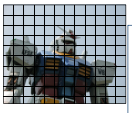
Shin Yoshizawa: shin@riken.jp

Poisson方程式の差分近似2

$$\Delta I = I_{xx} + I_{yy} \approx I(x+1,y) + I(x-1,y) + I(x,y+1) + I(x,y-1) - 4I(x,y)$$

$$\mathrm{div}\,\nabla g = \Delta g \approx g(x+1,y) + g(x-1,y) + g(x,y+1) + g(x,y-1) - 4g(x,y)$$

✓ 画像では…



$\Delta I = \mathrm{div}\nabla g$

$\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

	$I(x,y-1)$	
$I(x-1,y)$	$I(x,y)$	$I(x+1,y)$
	$I(x,y+1)$	

✓ 疎な線形連立方程式の数値解法は沢山ある：
- 直接法：LU分解などTAUCS, UMFPACK, SuperLU, etc.
- 繰り返し法：ガウスザイデル法、PCBCG等→Numerical Recipe in C.

Shin Yoshizawa: shin@riken.jp

Poisson Image Editingの実装・アルゴリズム

✓ 超簡単！

Mixing Gradientsの計算：方向毎

$$\Delta I = \begin{cases} \mathrm{div}\,\nabla g & \text{if } \|\nabla g\| \geq \|\nabla h\| \\ \mathrm{div}\,\nabla h & \text{else} \end{cases}$$

PoissonIE.h
内solve()関数:

Shin Yoshizawa: shin@riken.jp

Poisson Image Editingの実装・アルゴリズム2

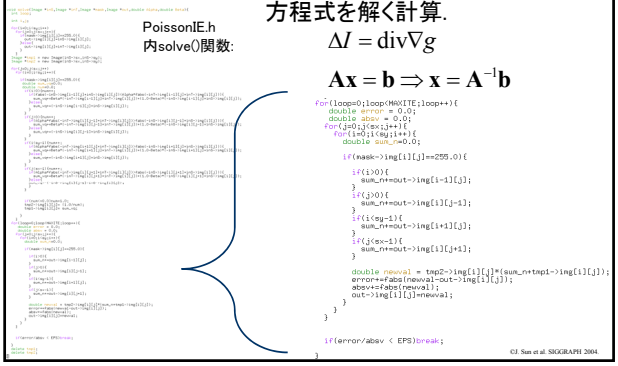
✓ 超簡単！

ガウスザイデル法によりPoisson方程式を解く計算.

$\Delta I = \mathrm{div}\nabla g$

$\mathbf{Ax} = \mathbf{b} \Rightarrow \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

PoissonIE.h
内solve()関数:



Shin Yoshizawa: shin@riken.jp

Poisson Matting

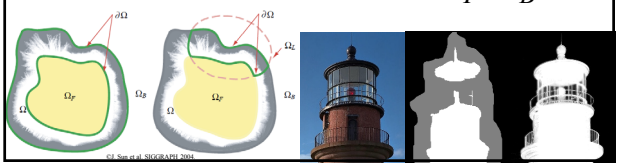
✓ Matte: 合成の為の透明度のマスク.

✓ Matting=Alpha-Blending: $I = \alpha F + (1-\alpha)B$

✓ Poisson Matting: Poisson Image Editingと同じ原理でMatteをPoisson方程式を解いて生成→非常に複雑な形状でもOK.

$$\nabla I = (F-B)\nabla\alpha + \alpha\nabla F + (1-\alpha)\nabla B \leftarrow (fg)' = f'g + fg'$$

$$\nabla\alpha \approx \frac{1}{F-B}\nabla I \qquad \Delta\alpha = \mathrm{div}\left(\frac{1}{F-B}\nabla I\right)$$

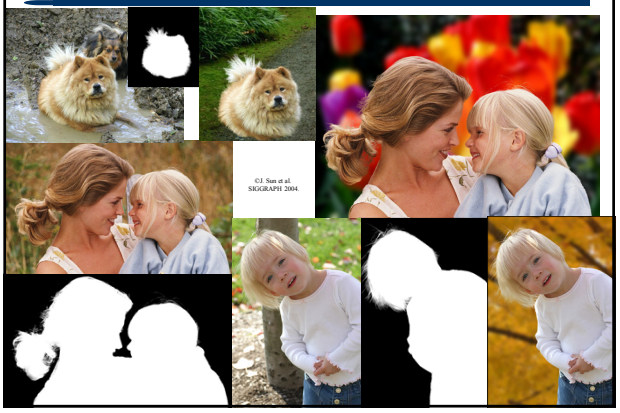


Shin Yoshizawa: shin@riken.jp

Poisson Matting2

Shin Yoshizawa: shin@riken.jp

Poisson Matting3



Shin Yoshizawa: shin@riken.jp

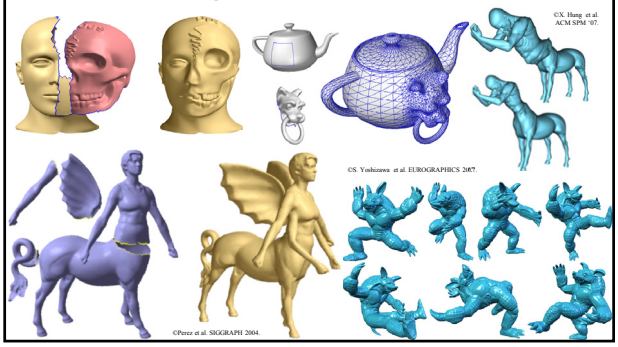
Poisson Matting4

✓ De-Fogging: 同じ原理でAlpha値の代わりにFoggのエフェクトをPoisson方程式を解く事で見積もる.

Shin Yoshizawa: shin@riken.jp

Poisson Mesh Editing

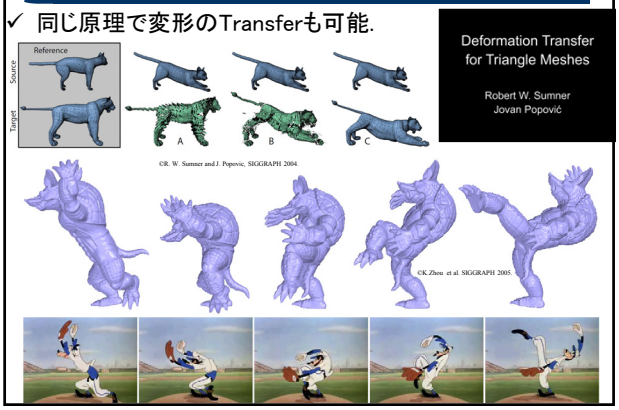
✓ 3次元形状・メッシュに拡張されてLaplacian-Poisson Mesh 変形・Editingと呼ばれる一大研究分野に発展.



Shin Yoshizawa: shin@riken.jp

Deformation Transfer

✓ 同じ原理で変形のTransferも可能.



Shin Yoshizawa: shin@riken.jp

演習:Poisson Image Editingを使ってみよう！

www.riken.jp/brict/Yoshizawa/Lectures/Lec10.pdf
www.riken.jp/brict/Yoshizawa/Lectures/Ex06.zip

Poisson Image Editingで画像合成:

1. Ex06内に用意されたプログラム群を動かしてみる.
   - ✓ Run_PoissonImageEditor.shを動かす.
2. MaskEditorを使って新しい合成を作ってみよう！
3. NumberEditorを使ってみる: Image AnalogyのTextureByNumbers用.

今日の演習は第3回レポートの内容なので
頑張ってくださいねーp(^^)q

Shin Yoshizawa: shin@riken.jp

演習:Ex06内の説明

www.riken.jp/brict/Yoshizawa/Lectures/Ex06.zip
www.riken.jp/brict/Yoshizawa/Lectures/Lec12.pdf

Ex06内の説明:コンパイルは端末で「make」Makefile

- ✓ PoissonIE.h: Poisson Image Editing(PIE)の本体.
  - PoissonImageEditor.cxx: PIEのメインソース.
  - LinearBlending.cxx: 線形合成プログラム.
- ✓ Ex06/MaskEditor: PIE用マスク作成GUI (Java).
- ✓ Ex06/NumberEditor: Image Analogy用TextureByNumbersのお絵かきGUI (Java).
- ✓ PIEとは関係ないファイル:
  - 前回までに使ったファイル:SimpleImage.h(画像クラス)、otsu.h(大津の二値化)、ppmio.h(カラー画像入出力)、pgmio.h(グレースケール画像入出力).

Shin Yoshizawa: shin@riken.jp

演習:PIE Blending

- ✓ PIEを実行するプログラム: 引数 6
  - PoissonImageEditor 入力Source画像.ppm 入力Mask画像.pgm 入力Target画像.ppm 出力合成画像.ppm 勾配倍率 Alpha(double>=0.0) **勾配Mix度** Beta(1.0>=double>=0.0)
  - Alpha(ターゲット画像勾配の倍率)とBeta(勾配のMix度合)は

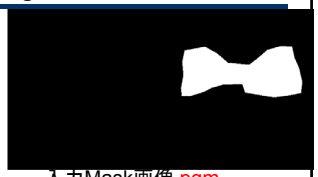
$$\Delta I = \begin{cases} \operatorname{div} \nabla g & \text{if } \|\nabla g\| \geq \alpha \|\nabla h\| \\ \beta \operatorname{div} \nabla h + (1-\beta) \operatorname{div} \nabla g & \text{else} \end{cases}$$

ターゲット画像の勾配　　ソース画像の勾配

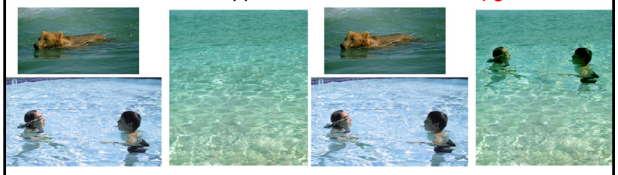
  - sh Run_PoissonImageEditor.shでも実行可能.
  - 注意点:入力Source画像.ppm、入力Mask画像.pgm、及び入力Target画像.ppmは全て同じサイズでないとダメ!

Shin Yoshizawa: shin@riken.jp

演習：Run_PoissonImageEditor.shの実行

入力Source画像.ppm　　入力Mask画像.pgm



入力Target画像.ppm　　出力合成画像.ppm

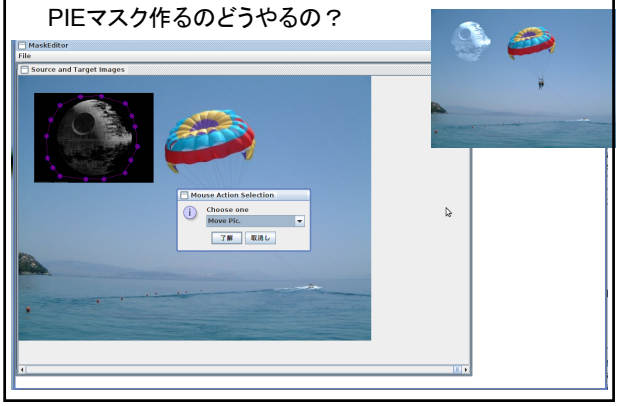
Shin Yoshizawa: shin@riken.jp

演習：Run_PoissonImageEditor.shの実行2

Shin Yoshizawa: shin@riken.jp

演習：MaskEditor

PIEマスク作るのどうやるの？



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」

Shin Yoshizawa: shin@riken.jp

演習：MaskEditor2

- ✓ PIE用マスク作成GUI (Java): Ex06/MaskEditor/
- ✓ コンパイル: 端末で「javac MaskEditor.java」
- ✓ 実行：端末で「java MaskEditor」
- ✓ sh Run_MaskEditor.shでもOK！

1. Source画像を読み込む: File->Load Source.　ppm画像
2. Target画像を読み込む:File->Load Target.　ppm画像
3. 左クリックでPolylineを生成して領域を作成(3点以上！).
4. Source画像の大きさと位置を合わせる.
   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. 右クリックでRemoveを選べばPolylineの頂点を削除可能.
   4. マウスの真ん中ホールで拡大縮小.
   5. Polylineの頂点は左クリックで移動可能.
   6. 下のスクロールバーで表示の透明度を変更可能.

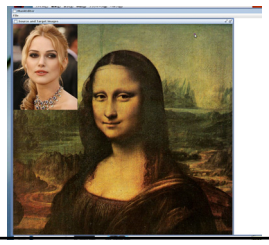
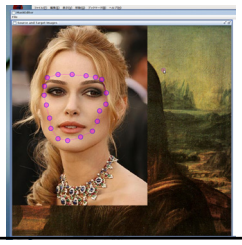
5. マスク画像(pgm)とTargetと同じ大きさのSource画像(ppm)の二つの画像をセーブ: File->Save Masks 注：セーブするファイル名に拡張子はいらない：ファイル名.pgmとファイル名.ppmが出来る.
6. PoissonImageEditorの第1，2引数へ、ターゲット画像は第3引数へ.

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！

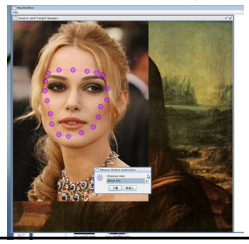



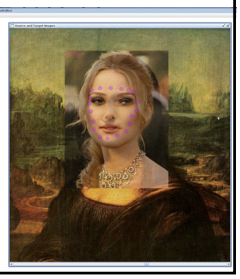
Shin Yoshizawa: shin@riken.jp

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

1. マスク画像(pgm)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という名前でセーブしてみよう！

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.pgmとファイル

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

Source　Mask　Target　合成結果

Shin Yoshizawa: shin@riken.jp

来週の予定

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

©Perez et al. SIGGRAPH 2003.

① 画像合成・Inpaintingその4

② 演習: レポート3の内容.