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-
アフィン変換と画素値の補間2

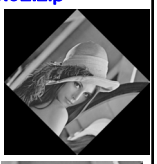
第６回講義
水曜日１限
教室6218情報処理実習室

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

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/Lec06.pdf
www.riken.jp/brict/Yoshizawa/Lectures/Ex02.zip
↑ Ex02.zipを保存→展開してください。



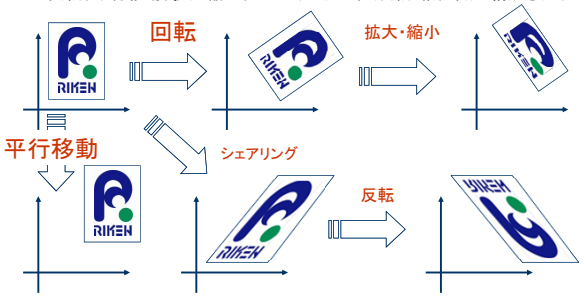
① アフィン変換の演習.
② 演習：前回の続き.

レポート第1回は今日〆切なので
頑張ってくださいねーp(^ ^)q！


Shin Yoshizawa: shin@riken.jp

復習：アフィン変換とは？

- アフィン変換(Affine Transformation)は既知の行列 $\mathbf{A}$ とベクトル $\mathbf{t}$ を用いて座標変換 $\mathbf{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{t}$ により点 $\mathbf{x}$ を点 $\mathbf{y}$ へ写像:
  - Collinearityを保存:直線は必ず直線に写像される.
  - 回転、平行移動、拡大縮小、シェアリングの組(反転・相似含)で構成される.



Shin Yoshizawa: shin@riken.jp

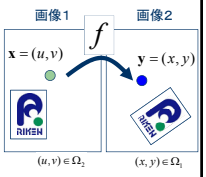
復習：アフィン変換とは？

- アフィン変換(Affine Transformation)は既知の行列 $\mathbf{A}$ とベクトル $\mathbf{t}$ を用いて座標変換 $\mathbf{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{t}$ により点 $\mathbf{x}$ を点 $\mathbf{y}$ へ写像:
- 2次元では、

$$\mathbf{y} = (x, y), \mathbf{x} = (u, v), \mathbf{t} = (b_1, b_2), \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix},$$

とすると、アフィン変換は以下の様に書ける.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$



$$f: \quad (u, v) \rightarrow (x, y)$$

- 厳密には平行移動がある形式は線形変換ではないが、斉次座標系で表すと線形変換となる.

Shin Yoshizawa: shin@riken.jp

Ex02内のファイル

- ppmio.h, pgmio.h, SimpleImage.h
- Scaling.cxx: ppm画像拡大縮小用ソースコード.
- Rotation.cxx: ppm画像回転用ソースコード.
- Shearing.cxx:　ppm画像シェアリング用ソースコード.
- affine.h: アフィン変換用関数(コレを編集).
- interpolation.h：補間用関数.

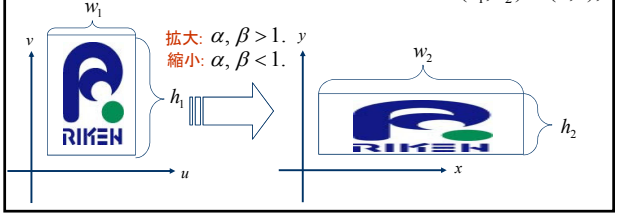
- Makefile: 端末にて「make」でコンパイル！
  - g++のオプションや実行ファイル名の指定、ソースコードのファイル名が記述してある.
  - emacsでMakefileを開いてみましょう！

Shin Yoshizawa: shin@riken.jp

復習：拡大・縮小1：Scaling

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

- X-スケーリング・ファクター: $\alpha = w_2 / w_1$.
- Y-スケーリング・ファクター: $\beta = h_2 / h_1$.
- 平行移動ベクトルはゼロベクトルになる: $\mathbf{t} = (b_1, b_2) = (0, 0)$,



Shin Yoshizawa: shin@riken.jp

プログラム例：拡大縮小

- ✓ 2次元配列A[2][2]にScaling行列をセットする関数を作ってみよう！

Ex02/affine.h内の…

$$A = \begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}$$

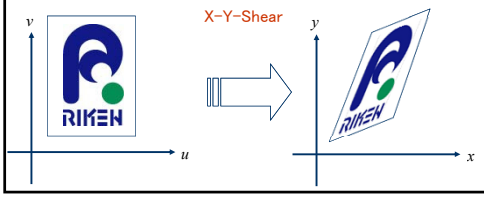
```
void Scaling(double A[2][2],double xfac, double yfac){
    A[0][0] = xfac;
    A[0][1] = A[1][0] = 0.0;
    A[1][1] = yfac;
}
```

Shin Yoshizawa: shin@riken.jp

復習：シェアリング3：Shearing

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & \alpha_s \\ \beta_s & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

- ✓ X-シェア・ファクター： α_s.
- ✓ Y-シェア・ファクター： β_s.
- ✓ 平行移動ベクトルはゼロベクトルになる: $\mathbf{t} = (b_1, b_2) = (0,0),$



Shin Yoshizawa: shin@riken.jp

プログラム例：シェアリング

- ✓ 2次元配列A[2][2]にShearing行列をセットする関数を作ってみよう！

Ex02/affine.h内の…

$$A = \begin{pmatrix} 1 & \alpha_s \\ \beta_s & 1 \end{pmatrix}$$

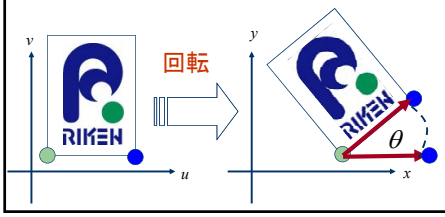
```
void Shearing(double A[2][2],double xfac, double yfac){
    A[0][0] = A[1][1] = 1.0;
    A[0][1] = xfac;
    A[1][0] = yfac;
}
```

Shin Yoshizawa: shin@riken.jp

復習：回転1：Rotation

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

- ✓ 回転行列： $A = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix},$
- ✓ 平行移動ベクトルはゼロベクトルになる: $\mathbf{t} = (b_1, b_2) = (0,0),$



- ✓ 回転角： $\theta,$

Shin Yoshizawa: shin@riken.jp

プログラム例：回転

- ✓ 2次元配列A[2][2]に回転行列をセットする関数を作ってみよう！
  - ✓ 注意1：cos(), sin()はmath.hをインクルードしてコンパイルのときには最後に「-lm」が必要.絶対値を取る関数fabs(), abs()も同様.
  - ✓ 注意2：math.hに入っている三角関数は弧度法(ラジアン)なので…

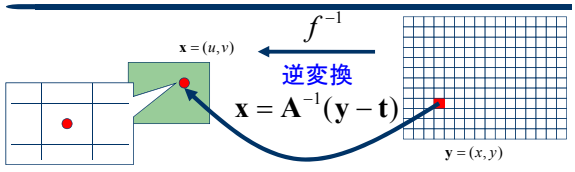
```
void Rotation(double A[2][2],double theta){
    double rad = (theta*PI)/180.0;
    A[0][0] = A[1][1] = cos(rad);
    A[0][1] = -sin(rad);
    A[1][0] = -A[0][1];
}
```

Ex02/affine.h内の…

$$A = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Shin Yoshizawa: shin@riken.jp

復習：順変換と逆変換4



- ✓ 入力と出力の画像サイズが違うとき:

$$\mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{c}_{out} - \mathbf{t}) + \mathbf{c}_{in},$$

逆変換:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \begin{pmatrix} x - c^{out}{}_x - b_1 \\ y - c^{out}{}_y - b_2 \end{pmatrix} + \begin{pmatrix} c^{in}{}_x \\ c^{in}{}_y \end{pmatrix}$$

Shin Yoshizawa: shin@riken.jp

演習：アフィン変換

✓ 中心で逆変換をする関数を作ってみよう！

Ex02/affine.h内に既にあります.

$$\mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{c}_{out} - \mathbf{t}) + \mathbf{c}_{in},$$

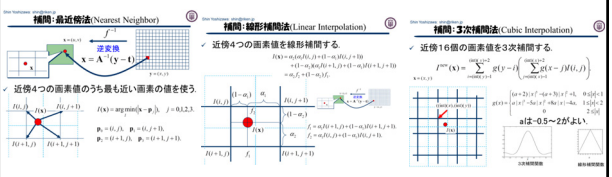
```
void AffineTransformation_BW(double inX[2],double A[2][2],
      double ci[2],double co[2], double t[2], double outX[2]){
    double tmp1[2],tmp2[2],tmp3[2];
    double mt[2]; mt[0] = -(co[0]+t[0]); mt[1] = -(co[1]+t[1]);
   Translate(inX,mt,tmp1);
    Vector_Matrix_Multiplication_Inverse(tmp1,A,tmp2);
    Translate(tmp2,ci,outX);
}
```

Shin Yoshizawa: shin@riken.jp

重要：補間法実装

✓ Ex02/interpolation.hに
- 最近傍法：NearestNeighbor(…)
- 線形補間法：LinearInterpolation(…)
- 3次補間法：CubicInterpolation(…)

が実装されているのでよく見ておいてください.



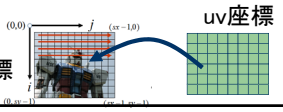
Shin Yoshizawa: shin@riken.jp

演習：実際に逆変換をしてみる！

✓ 各Scaling.cxx, Rotation.cxx, Shearing.cxxのmain内以下に注目！

変換後の画像サイズを計算する関数setNewSize()はEx02/affine.h内に既にあります.

```
double uv[2],xy[2];
for(i=0;i<out->sy;i++){ uv[1] = ((double)(i));
    for(j=0;j<out->sx;j++){ uv[0] = ((double)(j));
    AffineTransformation_BW(uv,A,t,centerIn,centerOut,xy);
    out->img[i][j] = CubicInterpolation(in,xy);
    }
}
```



Shin Yoshizawa: shin@riken.jp

演習：実際に逆変換をしてみる！

makeの後にScaling, Rotation, Shearingを動かしてみましょう！(「make clean」で実行ファイルを削除可能).

✓ ./Scaling 入力ppm 出力ppm xの倍率 yの倍率

例： ./Scaling lena.ppm lena_scale.ppm 0.75 2.5

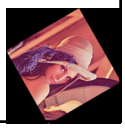
✓ ./Shearing 入力ppm 出力ppm xの比率 yの比率

例： ./Shearing lena.ppm lena_shearing.ppm 0.75 2.5

✓ ./Rotation 入力ppm 出力ppm 回転角度

例： ./Rotation lena.ppm lena_rotate.ppm 60.0

↑は3次スプライン補間を用いた逆変換でアフィン変換が実装されています.



Shin Yoshizawa: shin@riken.jp

演習の続き

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

✓ 前々回(Lec04.pdf)の演習4-1～4-5の続きを進めてください.

レポート第1回は今日〆切なので頑張ってp(^ ^)q！
遅れても提出可能でも減点(点数の0.8倍).

わかんなーい(´･ω･`)?という人は遠慮なく質問してください！

Shin Yoshizawa: shin@riken.jp

次回の予定

✓ 目標：大津法がプログラミング出来る様になる！

内容(7－9)：領域抽出
特に大津法・ラベリング