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

第５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

今日の授業内容

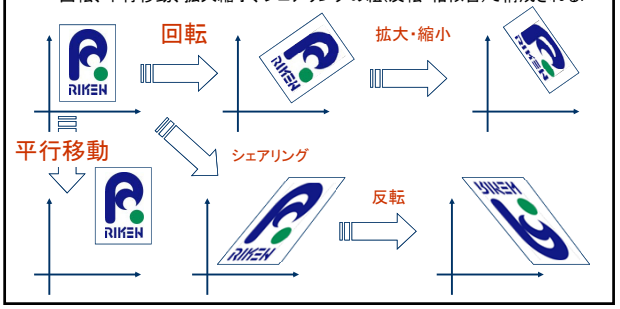
www.riken.jp/brict/Yoshizawa/Lectures/index.html

www.riken.jp/brict/Yoshizawa/Lectures/Lec05.pdf

① アフィン変換.
② 輝度値の補間.
③ 演習：前回の続き.

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.



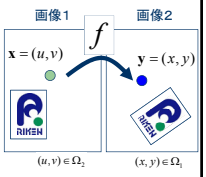
アフィン変換とは？

- 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.

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



- (u,v)に線形な変換である：厳密には平行移動がある形式は線形変換とは言わないが、斉次座標系で表すと線形変換となる.

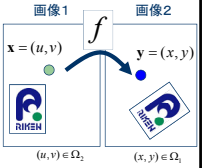
$$f:\ (u,v)\to(x,y)$$

斉次座標系での変換

- アフィン変換は斉次座標系では線形変換で表せる:

$$\begin{pmatrix}x\\ y\end{pmatrix}=\begin{pmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{pmatrix}\begin{pmatrix}u\\ v\end{pmatrix}+\begin{pmatrix}b_1\\ b_2\end{pmatrix}\Leftrightarrow\begin{pmatrix}x\\ y\\ 1\end{pmatrix}=\begin{pmatrix}a_{11} & a_{12} & b_1\\ a_{21} & a_{22} & b_2\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}u\\ v\\ 1\end{pmatrix}$$

$$\mathbf{y}=f(\mathbf{x})=\mathbf{A}\mathbf{x}+\mathbf{t}\Leftrightarrow\begin{pmatrix}\mathbf{y}\\ 1\end{pmatrix}=\begin{pmatrix}\mathbf{A} & \mathbf{t}\\ 0 & 1\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ 1\end{pmatrix}$$



- 斉次座標(Homogeneous coordinates): 以下の様に一次元高い空間で表す座標:

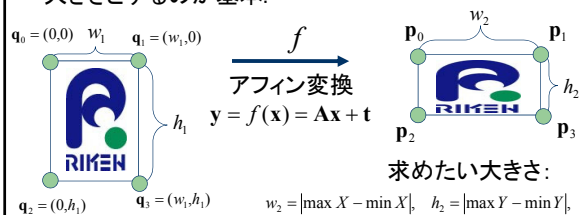
$$(x_1,x_2,\ldots,x_n)=\left(\frac{u_1}{u_{n+1}},\frac{u_2}{u_{n+1}},\ldots,\frac{u_n}{u_{n+1}}\right)$$

アフィン変換を行列積で表せたり，射影変換を行列で表せるためCGやCADでよく用いられる.

$$f:\ (u,v)\to(x,y)$$

重要: 変換後の画像の大きさ

- 変換後の画像の大きさは、まず入力画像の四隅の座標をアフィン変換してその横幅と縦の高さを新たな画像の大きさとするのが基本.



求めたい大きさ:

$$w_2=|\max X-\min X|,\quad h_2=|\max Y-\min Y|,$$

max,min: 4つのpの中で最も大きな/小さなX or Y座標.

$$\mathbf{p}_0=f(\mathbf{q}_0),\ \mathbf{p}_1=f(\mathbf{q}_1),\ \mathbf{p}_2=f(\mathbf{q}_2),\ \mathbf{p}_3=f(\mathbf{q}_3).$$

Shin Yoshizawa: shin@riken.jp

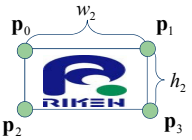
変換後の画像の大きさ2

max,min: 4つのpの中で最も大きな/小さなX or Y座標.

$$\mathbf{p}_0 = f(\mathbf{q}_0),\ \mathbf{p}_1 = f(\mathbf{q}_1),\ \mathbf{p}_2 = f(\mathbf{q}_2),\ \mathbf{p}_3 = f(\mathbf{q}_3).$$

$$\mathbf{p}_j = (x_j, y_j), \quad j = 0,1,2,3.$$ とすると、

$$\max X = \max(x_j), \quad j = 0,1,2,3.$$
$$\min X = \min(x_j, 0), \quad j = 0,1,2,3.$$
$$\max Y = \max(y_j), \quad j = 0,1,2,3.$$
$$\min Y = \min(y_j, 0), \quad j = 0,1,2,3.$$



変換後の画像の大きさ:

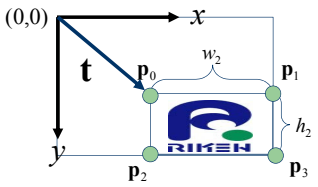
$$w_2 = |\max X - \min X|,$$
$$h_2 = |\max Y - \min Y|.$$

Shin Yoshizawa: shin@riken.jp

変換後の画像の大きさ3

✓ 平行移動も入っているときは平行移動ベクトルの大きさを最後に足す事に注意.

$$\mathbf{t} = (b_1, b_2)$$



最終的な画像の大きさ:

$$(w_2 + b_1, h_2 + b_2)$$

変換後の画像の大きさ:

$$w_2 = |\max X - \min X|,$$
$$h_2 = |\max Y - \min Y|.$$

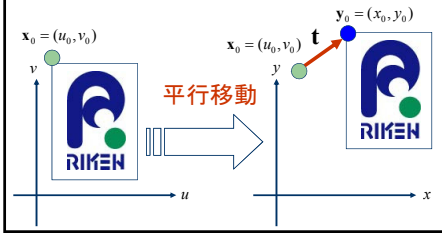
✓ 演習では変換後のサイズを計算する関数を提供します.

Shin Yoshizawa: shin@riken.jp

平行移動: Translation

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

✓ 平行移動ベクトル: $\mathbf{t} = (b_1, b_2) = \mathbf{y}_0 - \mathbf{x}_0 = (x_0 - u_0, y_0 - v_0),$



✓ 変換行列Aは単位行列になる:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Shin Yoshizawa: shin@riken.jp

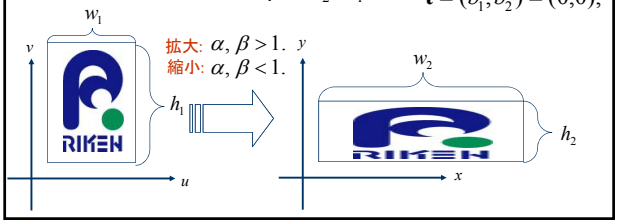
拡大・縮小1: Scaling

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

✓ 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

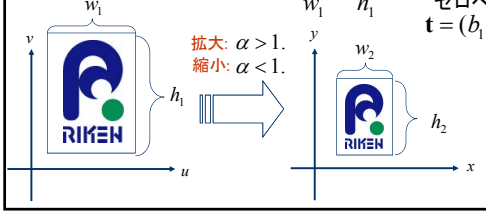
拡大・縮小2: Scaling

$\alpha = \beta$ のとき縦横比(Aspect Ratio)は保存される:

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

✓ スケーリング・ファクター: $\alpha = \frac{w_2}{w_1} = \frac{h_2}{h_1}.$

✓ 平行移動ベクトルはゼロベクトルになる: $\mathbf{t} = (b_1, b_2) = (0,0),$



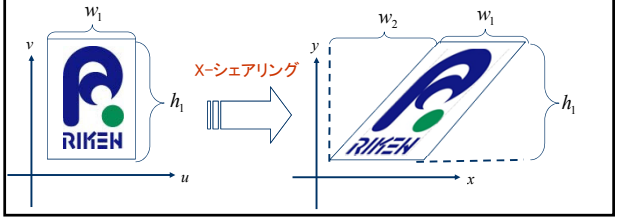
Shin Yoshizawa: shin@riken.jp

シェアリング1: X-Shearing

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & \alpha_s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

✓ X-シェア・ファクター: $\alpha_s = w_2 / h_1.$

✓ 平行移動ベクトルはゼロベクトルになる: $\mathbf{t} = (b_1, b_2) = (0,0),$

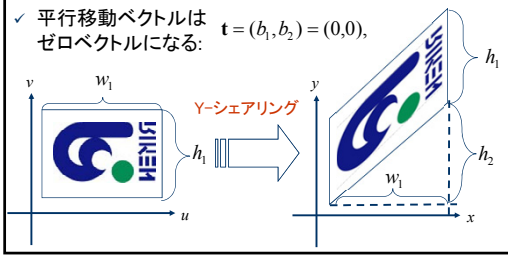


Shin Yoshizawa: shin@riken.jp

シェアリング2：Y-Shearing

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \beta_s & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

- ✓ Y-シェア・ファクター：$\beta_s = h_2 / w_1.$
- ✓ 平行移動ベクトルはゼロベクトルになる：$\mathbf{t} = (b_1, b_2) = (0,0),$

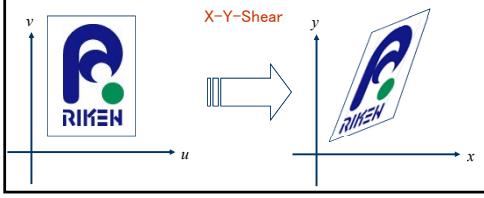


Shin Yoshizawa: shin@riken.jp

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

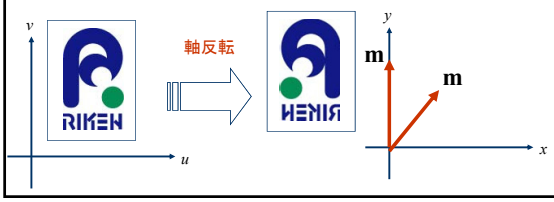


Shin Yoshizawa: shin@riken.jp

反転

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{|\mathbf{m}|^2} \begin{pmatrix} m_x^2 - m_y^2 & 2m_x m_y \\ 2m_x m_y & m_y^2 - m_x^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

- ✓ 原点を通る反転軸の方向ベクトル：$\mathbf{m} = (m_x, m_y).$
- ✓ 平行移動ベクトルはゼロベクトルになる：$\mathbf{t} = (b_1, b_2) = (0,0).$

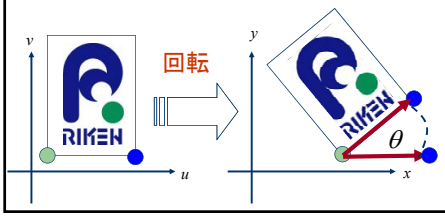


Shin Yoshizawa: shin@riken.jp

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：$\mathbf{t} = (b_1, b_2) = (0,0),$

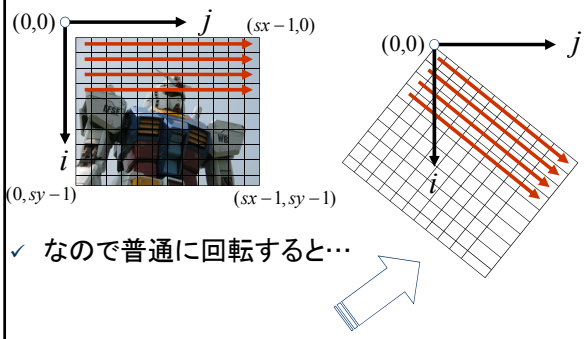


- ✓ 回転角：$\theta,$

Shin Yoshizawa: shin@riken.jp

回転2：Rotation

- ✓ 画像の座標系は…



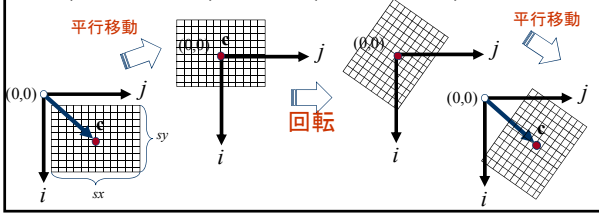
- ✓ なので普通に回転すると…

Shin Yoshizawa: shin@riken.jp

回転3：Rotation

- ✓ 画像の中心を原点とする座標系で回転させたい場合は一回、中心に平行移動させて回転後に逆向きの平行移動をする：$\mathbf{c} = (sx/2, sy/2)$

$$\mathbf{y} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \mathbf{x} \Rightarrow \mathbf{y} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} (\mathbf{x} - \mathbf{c}) + \mathbf{c}$$

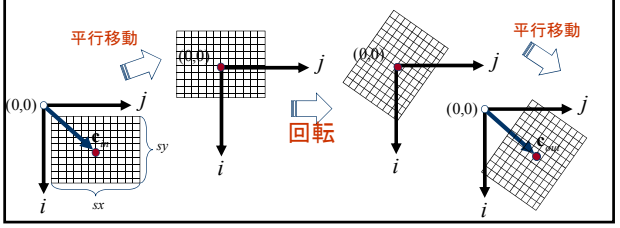


Shin Yoshizawa: shin@riken.jp

回転4：Rotation

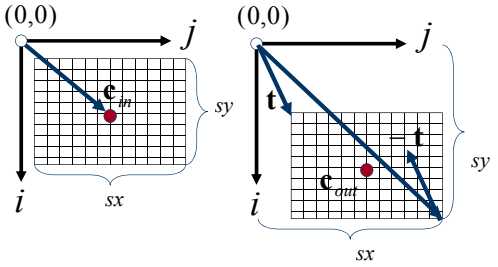
- 入力と出力の画像サイズが違うとき：

$$\mathbf{y} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}(\mathbf{x} - \mathbf{c}_{in}) + \mathbf{c}_{out}$$



回転5：Rotation

- 平行移動も入っているときは出力画像の中心位置$\mathbf{c}_{out}$が：(画像サイズ一平行移動ベクトル)の半分になる事に注意. $\mathbf{c}_{out} = ((sx, sy) - \mathbf{t})/2.$



写像の合成：複数の変換転

- 回転の後に拡大・縮小等、複数変換を行う場合は、画像を変換毎にセーブするのではなく、写像を合成して変換を行うのが基本：

$$\mathbf{y} = f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{t} \qquad f(\mathbf{x}) = f_1 \circ f_2 \circ \ldots \circ f_n(\mathbf{x})$$

$$A = \mu \underbrace{\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}}_{\text{回転}} \underbrace{\begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix}}_{\text{Y-シェア}} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix}}_{\text{Y-Dilatation}} \underbrace{\begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix}}_{\text{拡大・縮小}} \underbrace{\begin{pmatrix} 1 & \gamma \\ 0 & 1 \end{pmatrix}}_{\text{拡大・X-シェア}}$$

拡大・縮小　回転　Y-シェア　Y-Dilatation　拡大・縮小　Y-Dilatation　拡大・X-シェア

写像の分解1

- 行列分解の種類だけ分解可能：アフィン変換の行列は以下の9種類に分類される：

LDU分解：

Type 1: $A = \mu \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \gamma \\ 0 & 1 \end{pmatrix}$

Type 2: $A = \mu \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \gamma \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

Type 3: $A = \mu \begin{pmatrix} 1 & 0 \\ \gamma_y & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \gamma_x \\ 0 & 1 \end{pmatrix}$

Type 4: $A = \mu \begin{pmatrix} 1 & \gamma_x \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \gamma_y & 1 \end{pmatrix}$

写像の分解2

- 行列分解の種類だけ分解可能：アフィン変換の行列は以下の9種類に分類される：

QR分解：

Type 5: $A = \mu \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \gamma \\ 0 & 1 \end{pmatrix}$

Type 6: $A = \mu \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \gamma \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

QL分解：

Type 7: $A = \mu \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix}$

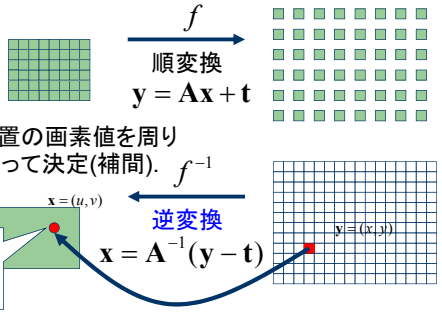
Type 8: $A = \mu \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

特異値分解：

Type 9: $A = \mu \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\lambda_x} \end{pmatrix} \begin{pmatrix} \lambda_x & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{pmatrix}$

重要：画像では？ 順変換と逆変換1

- 逆変換 + 画素値の補間によるアフィン変換：
  - 順写像(Forward Mapping) VS 逆写像(Backward Mapping)：

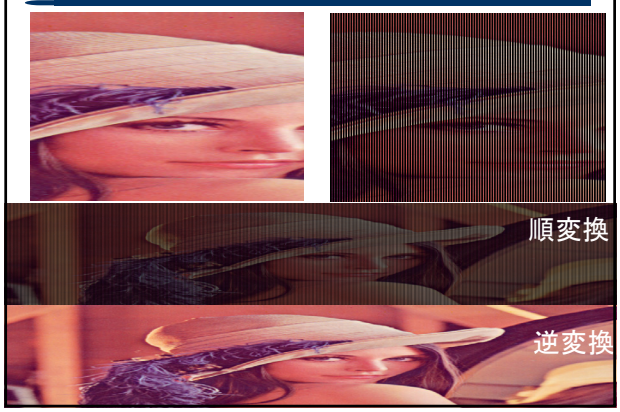


Shin Yoshizawa: shin@riken.jp

順変換と逆変換2

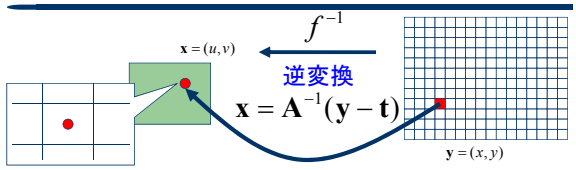
Shin Yoshizawa: shin@riken.jp

順変換と逆変換3



Shin Yoshizawa: shin@riken.jp

順変換と逆変換4



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

✓ ad−bc=0になる様な変換は作ってはいけない.

✓ ↓なので、

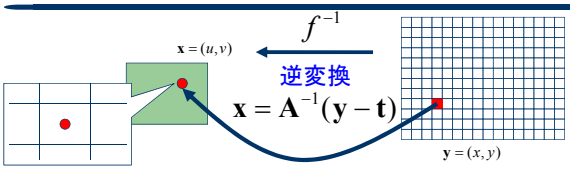
$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \Leftrightarrow A^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

逆変換:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \begin{pmatrix} x - b_1 \\ y - b_2 \end{pmatrix}$$

Shin Yoshizawa: shin@riken.jp

順変換と逆変換5



画像の中心を原点とする座標系では:

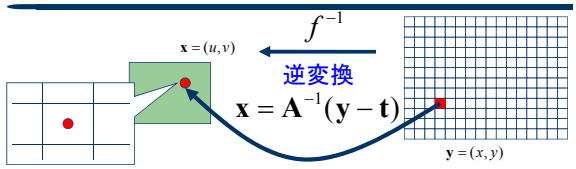
$$\mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{c} - \mathbf{t}) + \mathbf{c}, \quad \mathbf{c} = (c_x, c_y) = (sx/2, sy/2),$$

逆変換:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \begin{pmatrix} x - c_x - b_1 \\ y - c_y - b_2 \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$

Shin Yoshizawa: shin@riken.jp

順変換と逆変換6



✓ 入力と出力の画像サイズが違うとき:

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

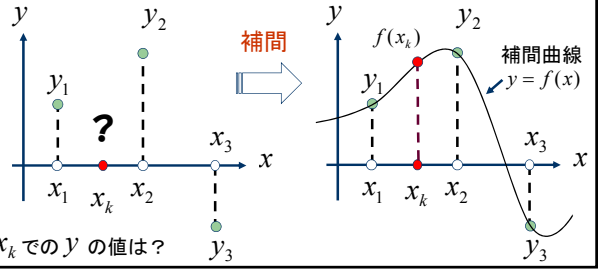
逆変換:

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

Shin Yoshizawa: shin@riken.jp

補間(Interpolation)とは？

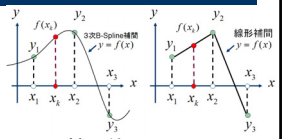
✓ 与えられた既知の数値データ列を基に、そのデータ列の間の数値(又は既知の数値データ列を通る関数)を求める事、内挿とも言う. 与えられたデータ外を考える場合は補外(外挿)と言う.



Shin Yoshizawa: shin@riken.jp

様々な補間法

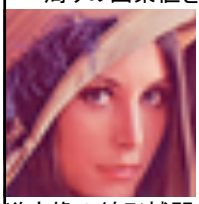
- ✓ 多くの方法がある:
  - ラグランジュ補間、多項式、
  - 線形補間(直線の式)、Sinc関数、
  - スプライン補間(B-Spline, etc.)、
  - RBF (Radial Basis Function)、エルミート補間等.
- ✓ 一番基本でよく用いられているのは線形補間と3次スプライン補間(Cubic Spline Interpolation).
- ✓ 補間はいろいろ使える.



Shin Yoshizawa: shin@riken.jp

重要:画像では？画素値の補間(2D)

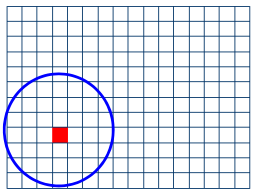
- ✓ 周りの画素値を使って補間:



逆変換＋線形補間

5倍拡大

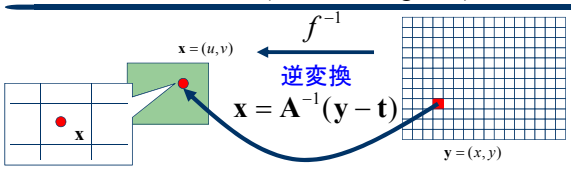
順変換



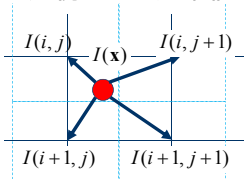
講義では…
最近傍法
線形補間法
3次Spline補間法

Shin Yoshizawa: shin@riken.jp

補間:最近傍法(Nearest Neighbor)



- ✓ 近傍4つの画素値のうち最も近い画素の値を使う.



$$I(\mathbf{x}) = \arg\min_{I}(|\mathbf{x}-\mathbf{p}_j|), \quad j=0,1,2,3.$$

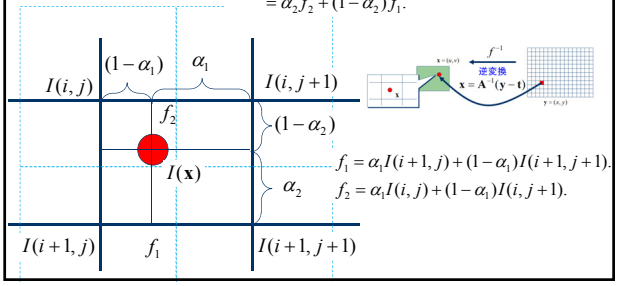
$$\mathbf{p}_0=(i,j), \quad \mathbf{p}_1=(i,j+1),$$
$$\mathbf{p}_2=(i+1,j), \quad \mathbf{p}_3=(i+1,j+1).$$

Shin Yoshizawa: shin@riken.jp

補間:線形補間法(Linear Interpolation)

- ✓ 近傍4つの画素値を線形補間する.

$$I(\mathbf{x}) = \alpha_2(\alpha_1 I(i,j)+(1-\alpha_1)I(i,j+1)) + (1-\alpha_2)(\alpha_1 I(i+1,j)+(1-\alpha_1)I(i+1,j+1)) = \alpha_2 f_2+(1-\alpha_2)f_1.$$

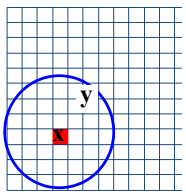


$$f_1=\alpha_1 I(i+1,j)+(1-\alpha_1)I(i+1,j+1).$$
$$f_2=\alpha_1 I(i,j)+(1-\alpha_1)I(i,j+1).$$

Shin Yoshizawa: shin@riken.jp

補間:線形畳み込み法

- ✓ 線形補間や3次Spline補間は「線形畳み込み(Linear Convolution)」と呼ばれる重み付和で計算出来る.重みの事を補間関数という.
- ✓ この方法は画像等の間隔が同じ格子が並んでいる場合に有効で簡単に適用出来る.



g とIの畳み込み:

補間したい画素の位置 $\mathbf{x}$　　既知の輝度値

$$I^{\text{new}}(\mathbf{x}) = \int g(\mathbf{x}-\mathbf{y})I(\mathbf{y})d\mathbf{y}$$

補間された輝度値　　補間関数

$\mathbf{x}$ 近傍の画素の位置 $\mathbf{y}$

Shin Yoshizawa: shin@riken.jp

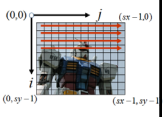
補間:線形畳み込み法

- ✓ 画像では、

$$I^{\text{new}}(\mathbf{x}) = \int g(\mathbf{x}-\mathbf{y})I(\mathbf{y})d\mathbf{y} = \iint g(x-u,y-v)I(u,v)dudv,$$

もしも、　$g(u,v)=g_1(u)g_2(v)$　ならば、

$$\begin{aligned} I^{\text{new}}(\mathbf{x}) &= \iint g(x-u,y-v)I(u,v)dudv \\ &= \iint g_1(x-u)g_2(y-v)I(u,v)dudv \\ &= \int g_2(y-v)\left(\int g_1(x-u)I(u,v)du\right)dv \\ &\approx \sum_i^N g_2(y-i)\left(\sum_j^M g_1(x-j)I[i][j]\right). \end{aligned}$$



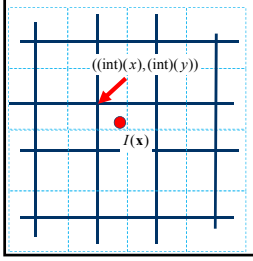
Shin Yoshizawa: shin@riken.jp

補間：3次補間法(Cubic Interpolation)

- ✓ 近傍16個の画素値を3次補間する.

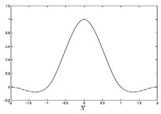
$$I^{\text{new}}(\mathbf{x}) = \sum_{i=(\text{int})(y)-1}^{(\text{int})(y)+2} g(y-i)\left(\sum_{j=(\text{int})(x)-1}^{(\text{int})(x)+2} g(x-j) I(i,j) \right).$$

$\mathbf{x} = (x, y)$

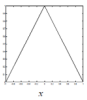


$$g(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1, & 0 \le |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a, & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}$$

aは−0.5〜2がよい.



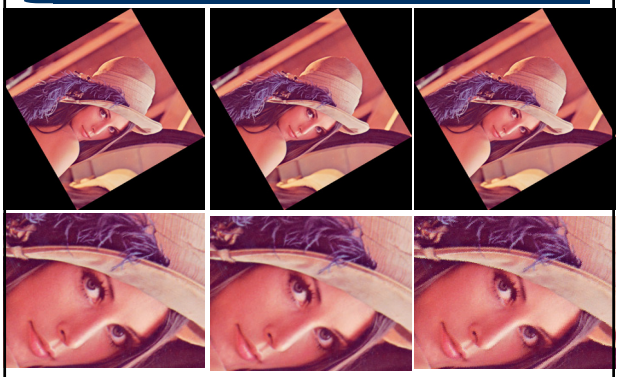
3次補間関数



線形補間関数

Shin Yoshizawa: shin@riken.jp

補間法の比較



3次補間　線形補間　最近傍補間

Shin Yoshizawa: shin@riken.jp

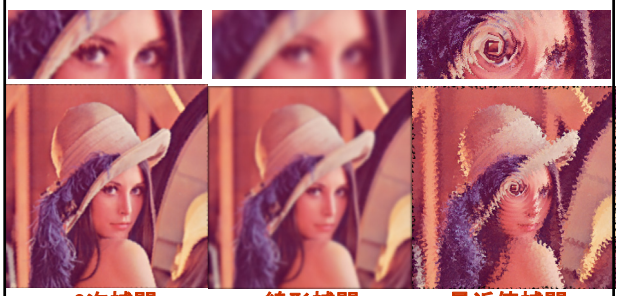
補間法の比較2

3次補間　線形補間　最近傍補間

Shin Yoshizawa: shin@riken.jp

補間法の比較3

- ✓ 3次補間、線形補間、最近傍の順に精度が良い.
- ✓ 例えば、5度づつ72回転しその都度補間を行うと...



3次補間　線形補間　最近傍補間

Shin Yoshizawa: shin@riken.jp

演習

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

アフィン変換・補間の演習は次週やります.

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

最初のレポートは↑の内容なので頑張ってp(^ ^)q！

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

レポート補足: gnuplot

- ✓ ヒストグラムのグラフ化方法：gnuplotというLinuxのコマンドを以下の様に使うと出来ます(xmgraceがインストールされてないので…).

1. 演習4-5ヒントにある様にfprintf()を用いて、x座標にビンのID(ビンの範囲の中央値などでもOK)、y座標に対応する頻度をテキストファイルにファイル名例えば **lena_hist_256.txt**として出力.
2. 授業のHPにあるHistogram_Example.pltをダウンロード:

http://www.riken.jp/brict/Yoshizawa/Lectures/Histogram_Example.plt

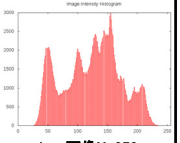
3. emacsでHistogram_Example.pltを編集.
4. 端末にて「gnuplot Histogram_Example.plt」を実行.

Shin Yoshizawa: shin@riken.jp

レポート補足2:gnuplot

- Histogram_Example.pltは講義HPにある二つのヒストグラムデータをpng画像にする設定です. 編集方法は、ビンの数がN、ヒストグラムデータのファイル名がhist_N.txt、出力したいpngファイル名をoutput_hist_N.pngとすると、

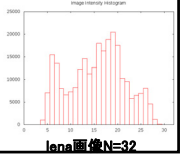
```
set xrange[0:256]
set output "lena_hist_256.png"
plot "lena_hist_256.txt" with boxes

set xrange[0:32]
set output "lena_hist_32.png"
plot "lena_hist_32.txt" u 1:2 with boxes
```



lena画像N=256

↑の部分を以下の様に編集すればOK.

```
set xrange[0:N]
set output "lena_hist_N.png"
plot "lena_hist_N.txt" u 1:2 with boxes
```



lena画像N=32

Shin Yoshizawa: shin@riken.jp

次回の予定

- アフィン変換・補間の演習をやります.