# 第6回　幾何学的変換

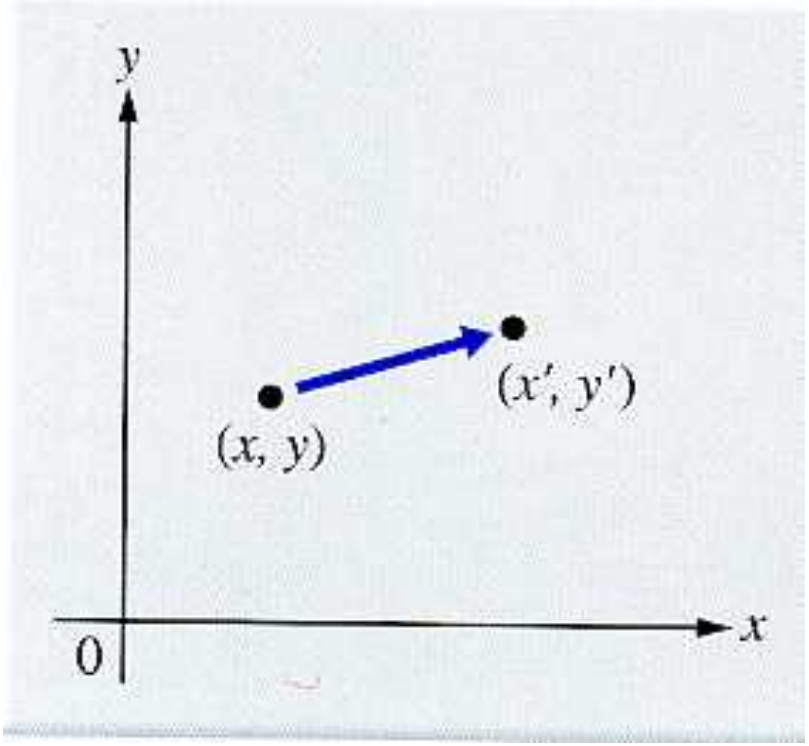
１．線形変換、アフィン変換など
２．補間
３．イメージモザイキング（応用）

# 線形変換(Linear Transformation)

- 座標(x,y)の位置の点が、変換により座標(x’,y’)に移動するとき、次式で表され、線形変換と呼ぶ。

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

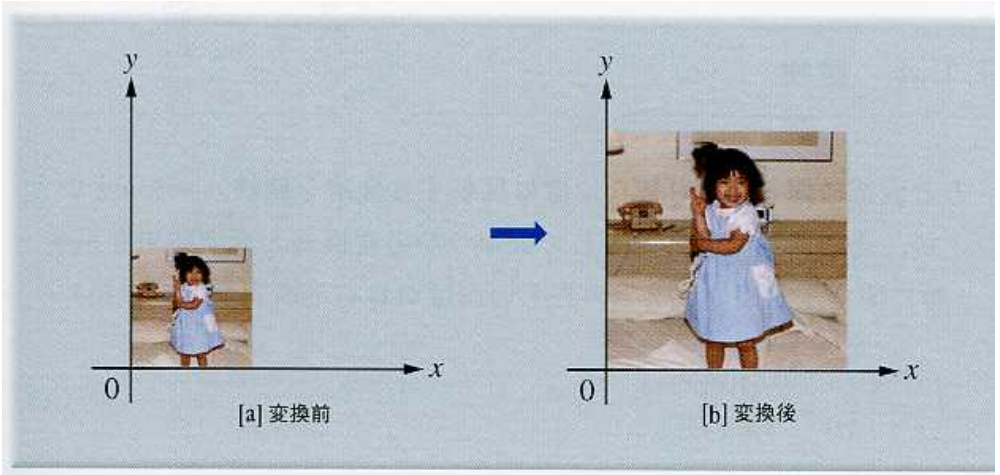
ここで、$a,b,c,d$は任意の実数



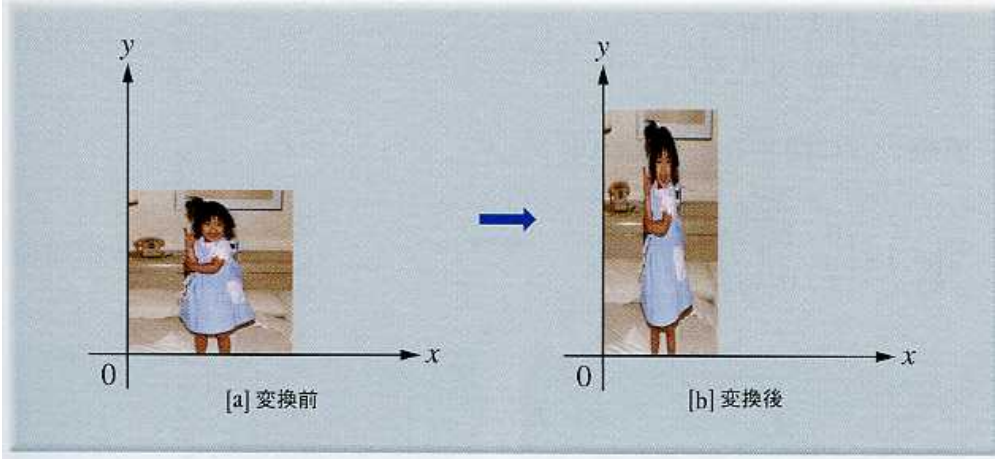
# 拡大・縮小

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- ここで、sx,syは、それぞれx方向、y方向の拡大（または縮小）率で、その値が1より大きいときは拡大、1より小さいときは縮小となる




■図9.2——拡大・縮小（$s_x = s_y = 2$）

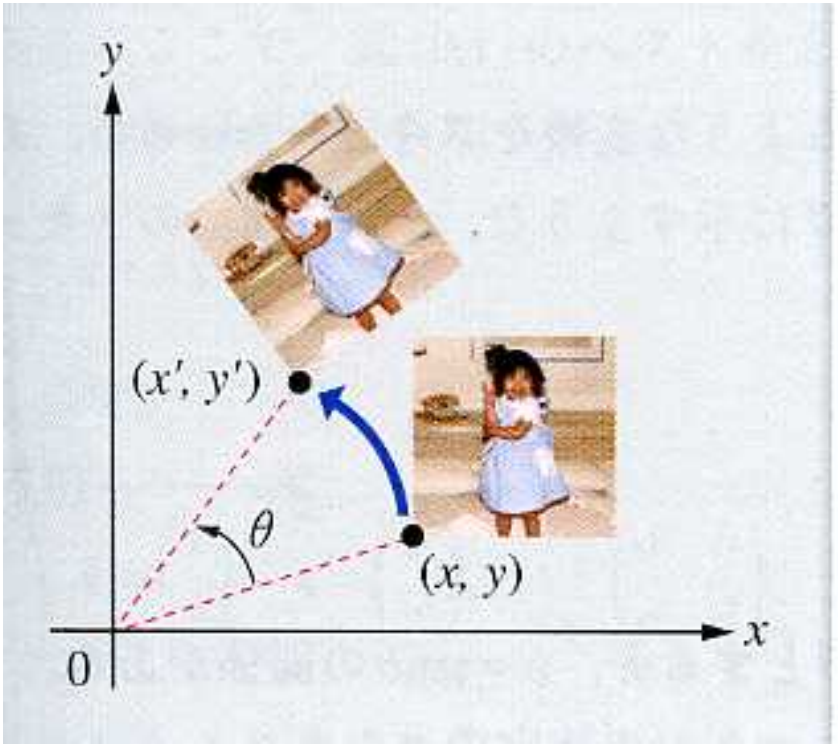



■図9.3——拡大・縮小（$s_x = 0.75$, $s_y = 1.5$）

# 回転(rotation)

- 原点を中心に、反時計周りに角度θだけ回転する変換:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

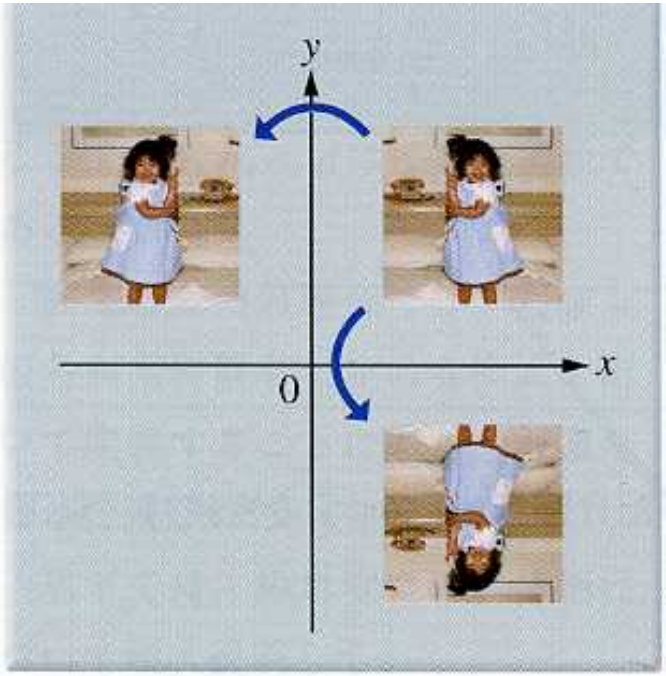


# 鏡映（reflection）

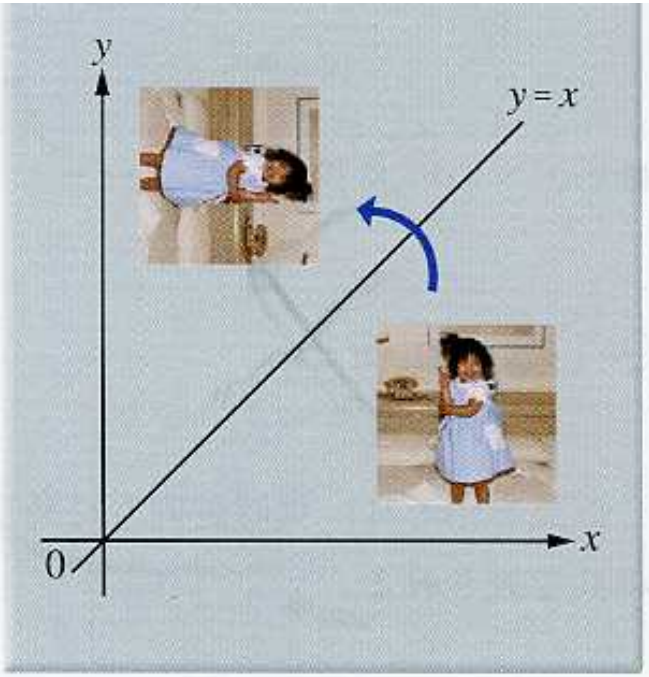
- ある直線に関して、対称な位置に反転する変換を鏡映という。

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}; \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}; \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

**x軸に関する変換；　　y軸に関する変換；　　直線y=xに関する変換**




■図9.5——鏡映変換（$x$ 軸, $y$ 軸）




■図9.6——鏡映変換（$y=x$）

# スキュー（skewing）

- **長方形を傾けて平行四辺形にするような変換をスキュー(skewing)、あるいはせん断（shearing)という。**

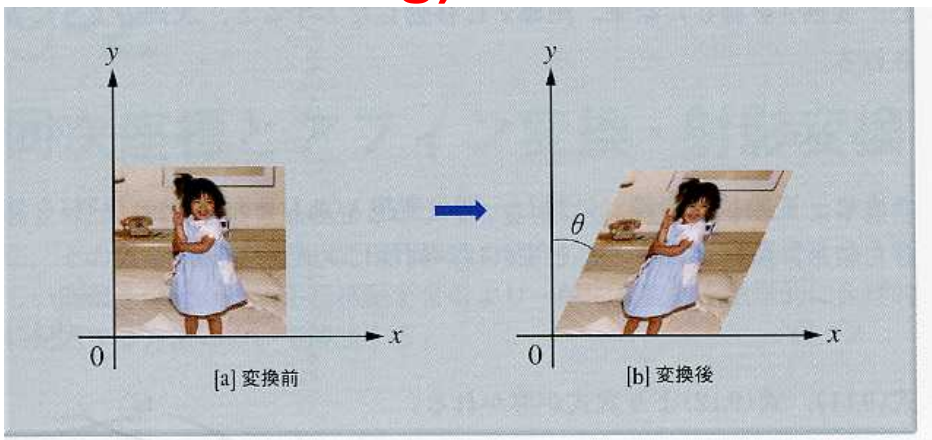
$x$軸方向のスキュー

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$

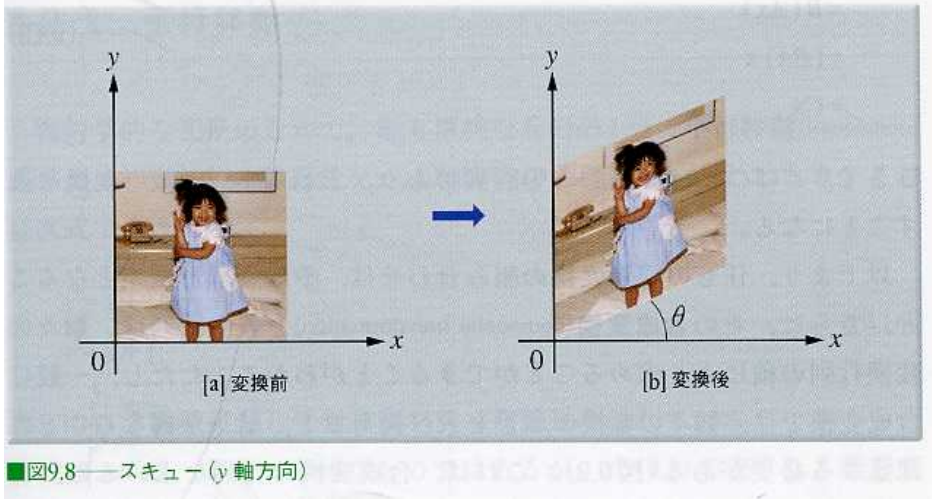
$y$軸方向のスキュー

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ b & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix};$$

$$b = \tan\theta$$




■図9.7——スキュー（$x$ 軸方向）




■図9.8——スキュー（$y$ 軸方向）

# 合成変換

これまでの線形変換はベクトルと行列で表すと

$$\mathbf{x}' = \mathbf{A}\mathbf{x}$$

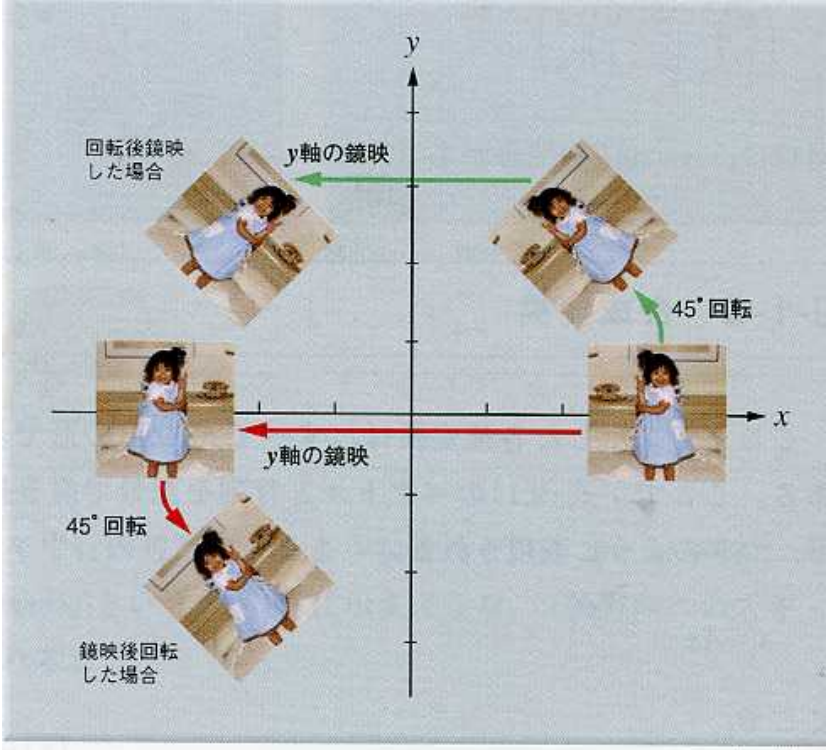
$$where\ \mathbf{x}' = \begin{pmatrix} x' \\ y' \end{pmatrix}; \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}; \mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

**複数変換がある場合、**

$$\mathbf{x}' = \mathbf{A}\mathbf{x}$$

$$\mathbf{x}'' = \mathbf{B}\mathbf{x}' = \mathbf{B}(\mathbf{A}\mathbf{x}) = (\mathbf{B}\mathbf{A})\mathbf{x}$$

$$= \mathbf{C}\mathbf{x}$$




■図9.9——順番を入れ換えると合成変換が異なる例

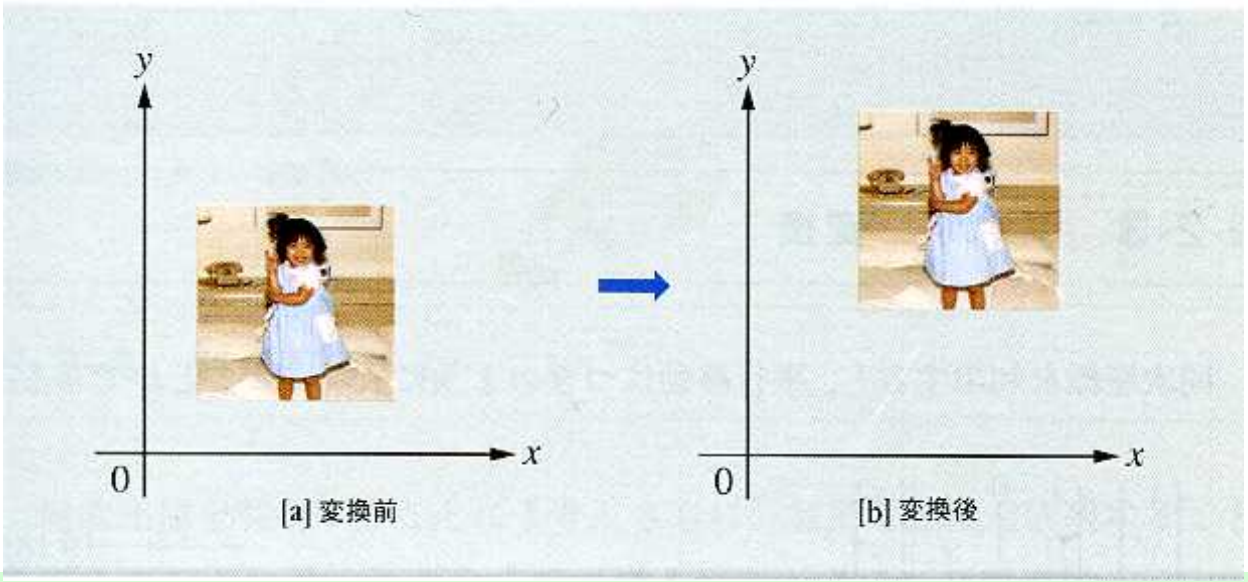
任意の線形変換の組み合わせはやはり線形変換である。
その合成変換を表す行列は個々の変換行列の積になる。

# 平行移動(translation)

- X軸方向の移動量を$t_x$, y軸方向の移動量を$t_y$とすると、

$$x' = x + t_x$$

$$y' = y + t_y$$




平行移動は、線形変換で表現できないので、平行移動を含むような合成変換では、行列の掛け算で表現することができない

⇩

新しい座標系の導入

# 同次座標（homogeneous coordinates）

- **同次座標**：座標（x,y)に対し、その要素数を次のように一つ増やした座標（$\xi_1,\xi_2,\xi_3$）

$$x = \frac{\xi_1}{\xi_3}; \quad y = \frac{\xi_2}{\xi_3}$$

- 同次座標においては、$\lambda \neq 0$なる任意の$\lambda$にたいして、（ $\xi_1,\xi_2,\xi_3$ ）と（ $\lambda\xi_1,\lambda\xi_2,\lambda\xi_3$ ）は同じ点を表している。すなわち、同次座標による表現では、定数倍をしても変わらない。このような関係を同値（equivalent)であるという。

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} \sim \begin{pmatrix} \lambda\xi_1 \\ \lambda\xi_2 \\ \lambda\xi_3 \end{pmatrix}$$

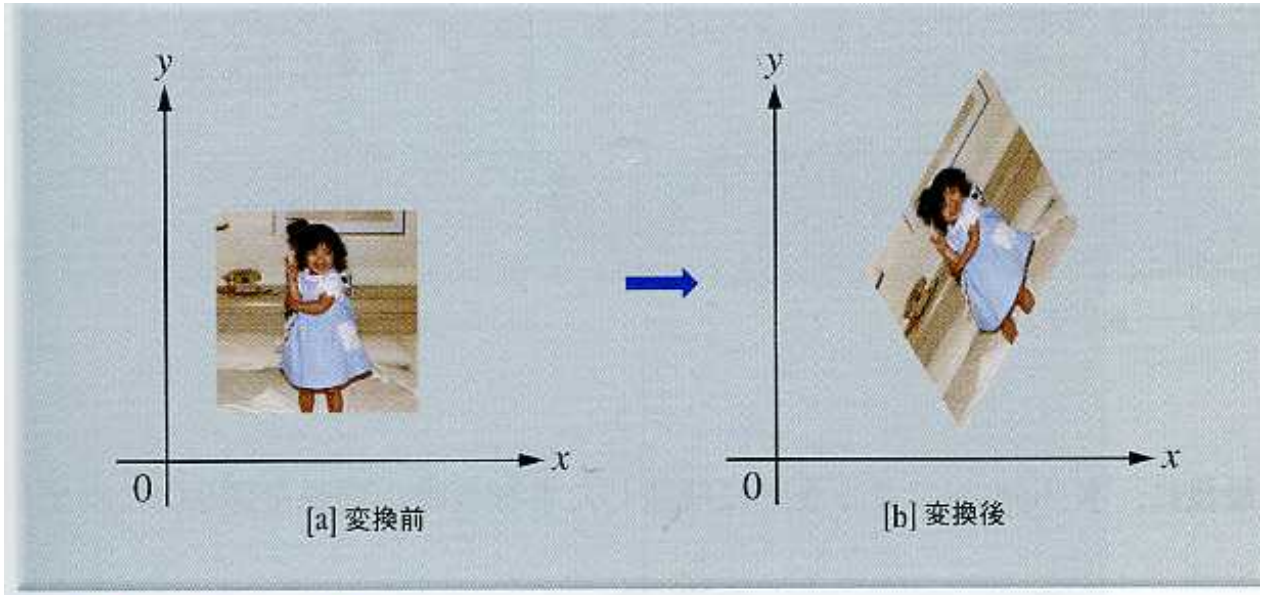
# アフィン変換（affine transformation）

- 同次座標において、平行移動と線形変換は次のように表される。

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}; \quad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- 任意の線形変換と平行移動を組み合わせた変換を**アフィン変換**とよぶ。

$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{pmatrix}$$




■図9.11——アフィン変換

- アフィン変換において、原図形の長さと角度は保たれないが、線分の直線性と平行性は保たれる。
- ユークリッド変換（Euclidean transform): 任意の回転と平行移動の組み合わせ（アフィン変換の特殊な例）

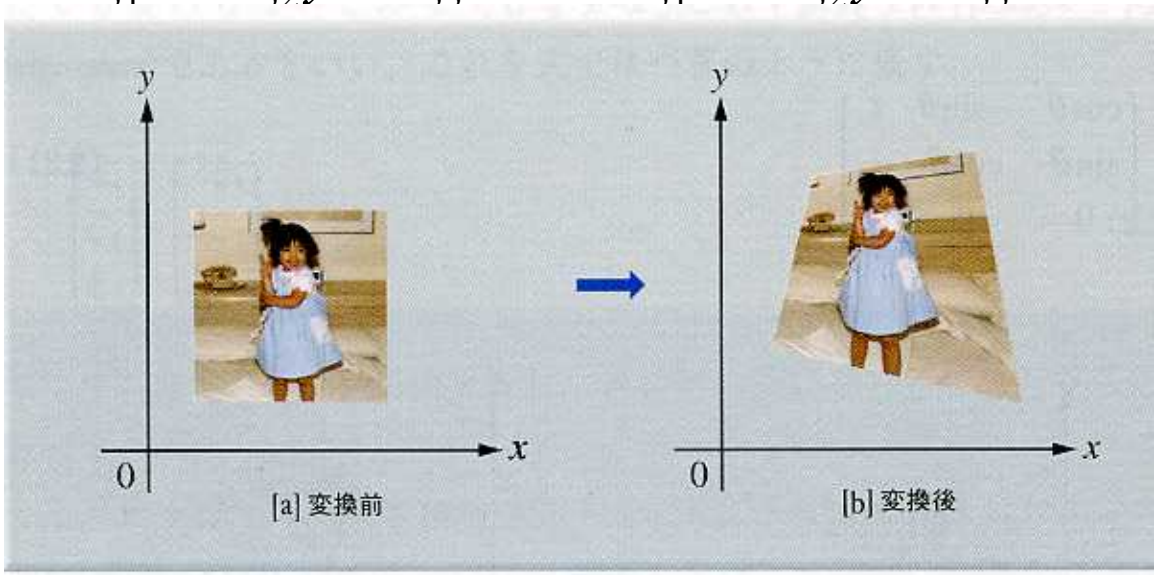
$$\begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

# 射影変換(Projective transform)

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

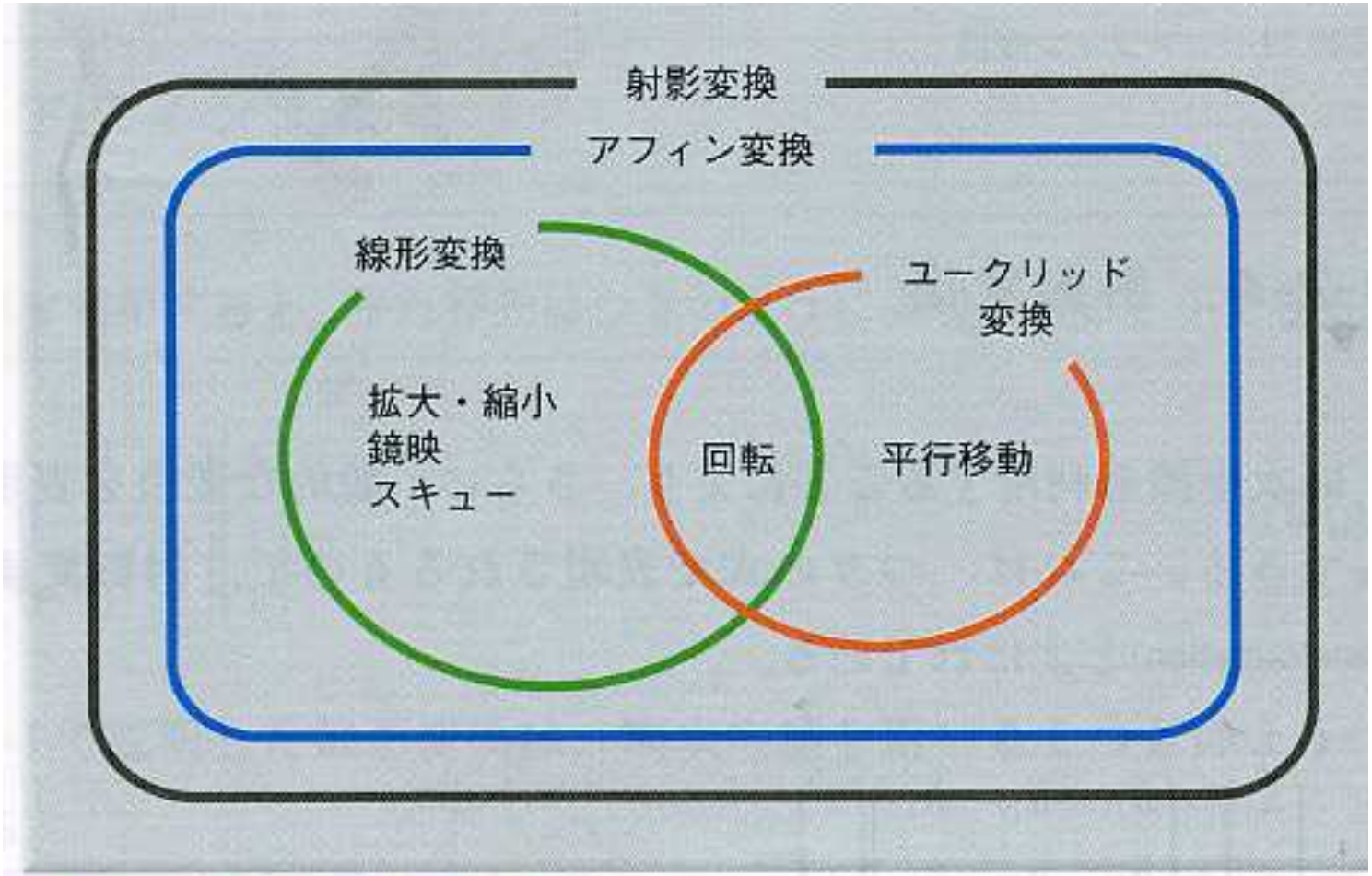
**射影変換において、線分の直線性は保たれるものの、平行性は失われる。**

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}; \; y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$




■図9.12——射影変換

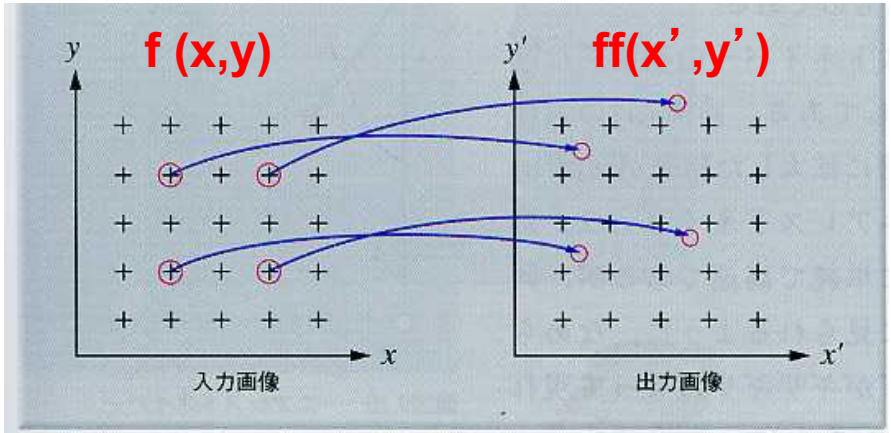
# 幾何学変換の関係



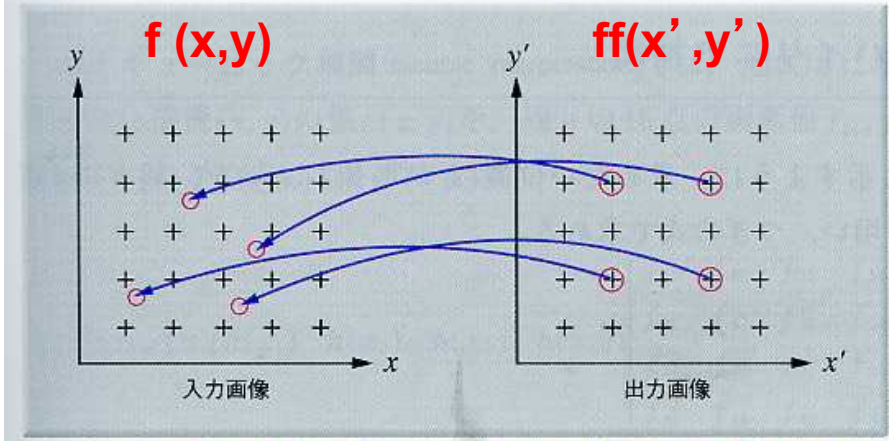

■図9.13——幾何学的変換の関係

# 逆変換と再標本化

**ディジタル画像の座標は整数である。**
**変換後の座標は整数ではないので、整数になるように再標本化する必要がある。**




■図9.14——幾何学的変換後の位置（図中+は，それぞれの画像における画素位置（標本化位置）を表す）




■図9.15——逆変換と再標本化（図中+は，それぞれの画像における画素位置（標本化位置）を表す）

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}^{-1} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

# 再標本化のステップ

(1) 変換後の出力画像におけるある画素位置に対し、適用する幾何学的変換の逆変換を行い、元の入力画像に対する位置を求める。逆変換は、変換が行列で表現されているので、その逆行列を求める。

(2) 上で求めた入力画像上の位置は、一般に入力画像の画素位置からずれている。そこで、その位置の値を補間より、周りの画素位置の値を利用して求める。

(3) 上の処理を、出力画像上のすべての画素位置に対して行う。

# 補間（interpolation)

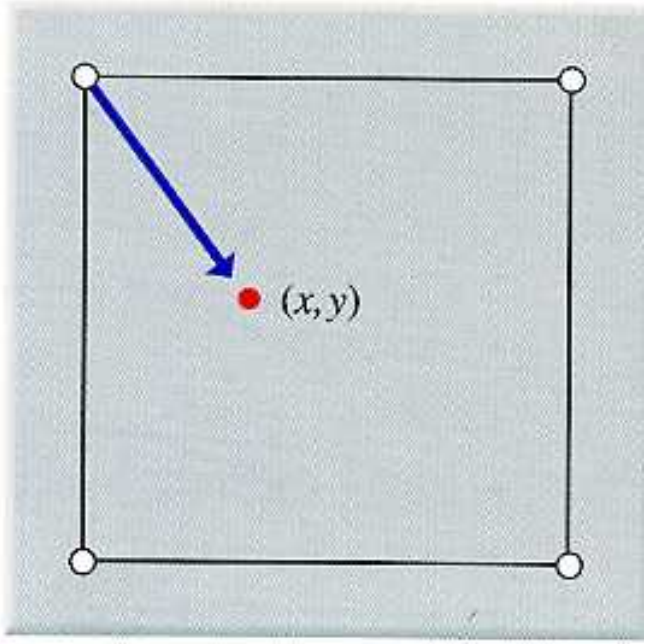
- ニアレストネイバー（nearest neighbor)
- バイリニア補間（bilinear interpolation)
- バイキュービック補間（bicubic interpolation)

# ニアレストネイバー（nearest neighbor)

- 求めたい位置に最も近い画素位置の値をそのまま利用する

$$ff(x', y') = f([x + 0.5], [y + 0.5])$$

ここで、f(i,j)は、入力画像の位置(i,j)の画素値を、記号[ ]はガウス記号で、[ ]内の値を超えない最大の整数値を返す。




例えば：x=1.2, y=2.6
[x+0.5]=[1.7]=1
[y+0.5]=[3.1]=3

四捨五入

ff(x’,y’)=f(1,3)

ニアレストネイバーは、処理が非常に単純で高速であるが、滑らかなエッジがギザギザになって現れるジャギーが発生しやすい。

# ニアレストネイバーのアルゴリズム

for (x’ =0; x’ <N; x’ ++)

{

for (y’ =0; y’ <M, y’ ++)

{

$$x = x'\cos\theta + y'\sin\theta;$$

$$y = -x'\sin\theta + y'\cos\theta;$$

$$ff(x', y') = f(\mathrm{int}(x + 0.5), \mathrm{int}(y + 0.5));$$

}

}

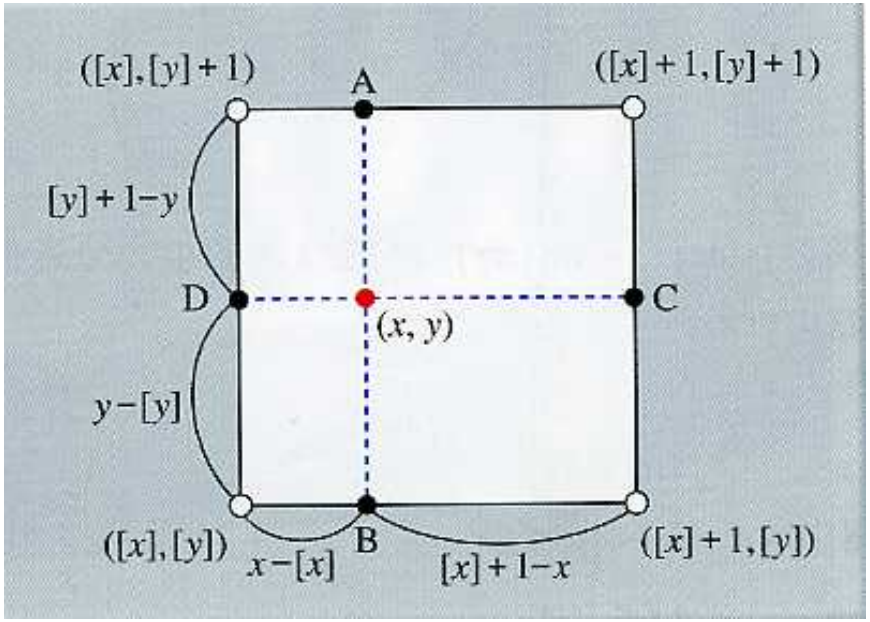
# バイリニア補間（bilinear interpolation）

- **求めたい位置（x,y)の値I(x,y)を、周りの4点の画素値を用い、次の式で求める。**

$$I(x,y) = \begin{pmatrix} [x]+1-x & x-[x] \end{pmatrix} \begin{pmatrix} f([x],[y]) & f([x],[y]+1) \\ f([x]+1,[y] & f([x]+1,[y]+1) \end{pmatrix} \begin{pmatrix} [y]+1-y \\ y-[y] \end{pmatrix}$$

$$= ([x]+1-x)([y]+1-y)f([x],[y]) + ([x]+1-x)([y]-y)f([x],[y]+1)$$
$$+ ([x]-x)([y]+1-y)f([x]+1,[y]) + ([x]-x)([y]-y)f([x]+1,[y]+1)$$

**バイリニア補間では、周りの4点の画素値の重み付の平均値を求めることになるため、平滑化の効果が生じる。そのため、ニアレストネイバーのようなジャギーが目立たなくなるが、エッジがなまってしまう傾向がある。**

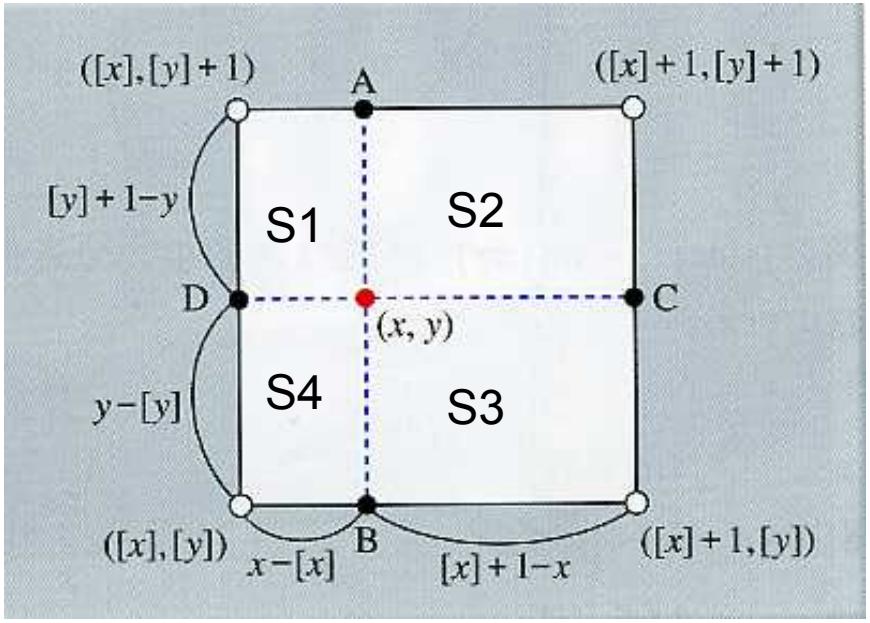


# バイリニア補間（bilinear interpolation)

$$S_1 = (x-[x])([y]+1-y); \quad S_2 = ([x]+1-x)([y]+1-y)$$

$$S_3 = ([x]+1-x](y-[y]); \quad S_4 = (x-[x])(y-[y])$$

$$ff(x',y') = f(x,y) = S_3 \cdot f([x],[y]+1) + S_4 \cdot f([x]+1,[y]+1)$$
$$+ S_1 \cdot f([x]+1,[y]) + S_2 \cdot f([x],[y])$$




**例えば：x=1.2, y=2.6**
**[x]=[1.2]=1, [y]=[2.6]=2**

**S1=0.2*0.4=0.08; S2=0.8*0.4=0.32**
**S3=0.8*0.6=0.48; S4=0.2*0.6=0.12**

**ff(x’,y’)=0.48*f(1,3)+0.12*f(2,3)**
**+0.08*f(2,2)+0.32*f(1,2)**

# バイリニア補間のアルゴリズム

for (x’=0; x’<N; x’++)

{

for (y’=0; y’<M, y’++)

{

$$x = x'\cos\theta + y'\sin\theta;$$

$$y = -x'\sin\theta + y'\cos\theta;$$

$$S_1 = (x - [x])([y] + 1 - y);$$

$$S_2 = ([x] + 1 - x)([y] + 1 - y);$$

$$S_3 = ([x] + 1 - x](y - [y]);$$

$$S_4 = (x - [x])(y - [y]);$$

$$ff(x', y') = S_3 \cdot f([x],[y]+1) + S_4 \cdot f([x]+1,[y]+1) + S_1 \cdot f([x]+1,[y]) + S_2 \cdot f([x],[y]);$$

}

}

# バイキュービック補間（bicubic interpolation)

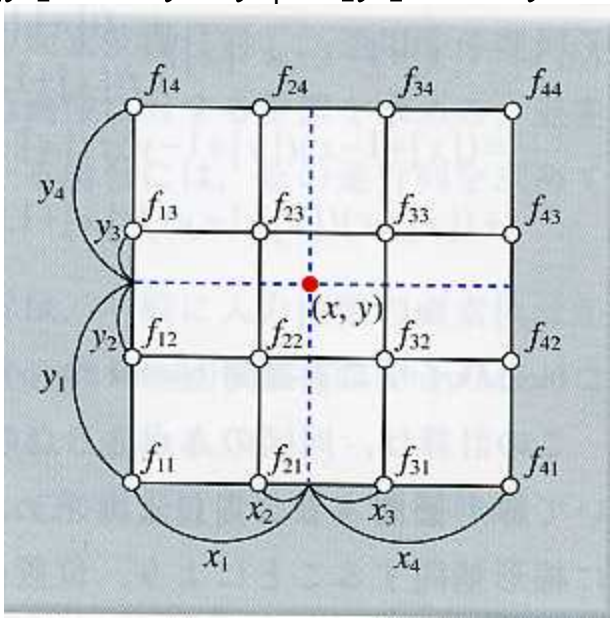
- **求めたい位置（x,y)の値I(x,y)を、周りの16点の画素値$f_{11}$, $f_{12}$,…,$f_{44}$を用い、次の式で求める。**

$$I(x,y)=\begin{pmatrix}h(x_1) & h(x_2) & h(x_3) & h(x_4)\end{pmatrix}\begin{pmatrix}f_{11} & f_{12} & f_{13} & f_{14}\\ f_{21} & f_{22} & f_{23} & f_{24}\\ f_{31} & f_{32} & f_{33} & f_{34}\\ f_{41} & f_{42} & f_{43} & f_{44}\end{pmatrix}\begin{pmatrix}h(y_1)\\ h(y_2)\\ h(y_3)\\ h(y_3)\end{pmatrix}$$

$$where\ \ x_1=1+x-[x];\ x_2=x-[x];\ x_3=[x]+1-x;\ x_4=[x]+2-x$$

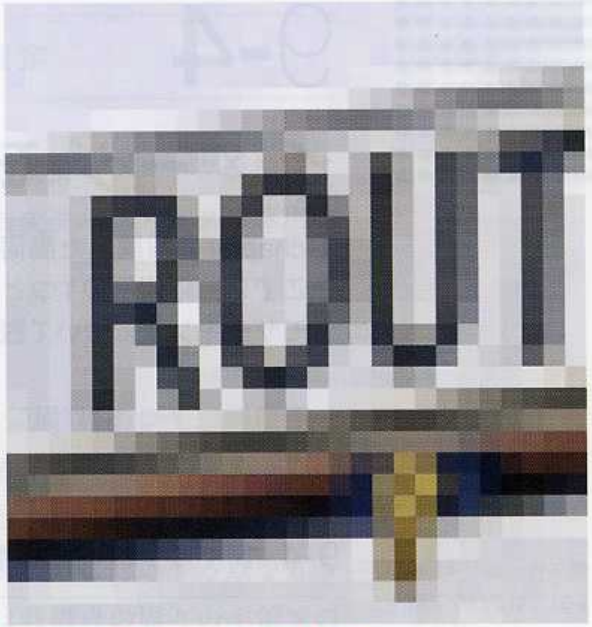
$$y_1=1+y-[y];\ y_2=y-[y];\ y_3=[y]+1-y;\ y_4=[y]+2-y$$

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

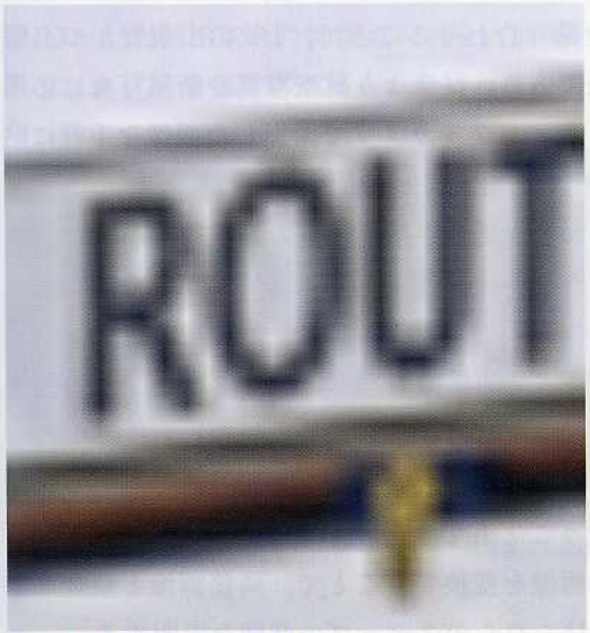



**バイリニア補間に比べ、よりシャップで自然な画像が得られる。**

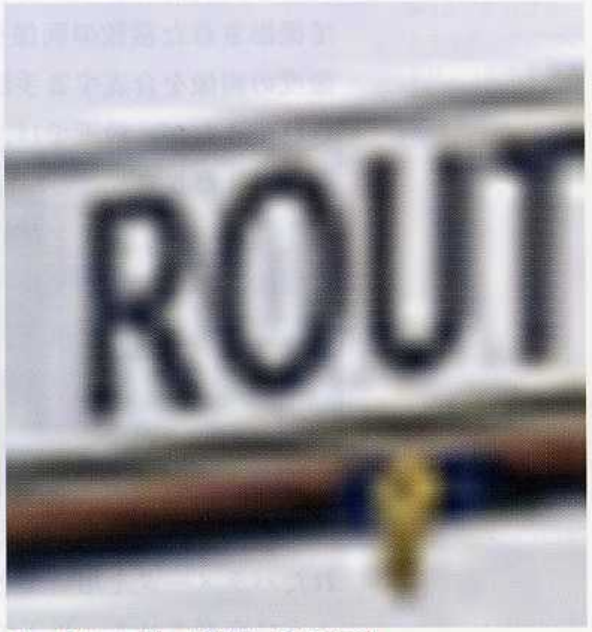

[a] 原画像




[b] ニアレストネイバーによる拡大画像




[c] バイリニア補間による拡大画像




[d] バイキュービック補間による拡大画像

■図9.19——各種補間法を利用した画像の拡大結果（[a]の赤い四角の部分）

# プログラミング課題

- 画像（Female.pgm)を$\theta$=45度を回転するプログラムを作成しなさい。

  以下の補間方法をそれぞれ用い、それぞれの結果を出力しなさい。

  ニアレストネイバー（nearest neighbor)

  バイリニア補間（bilinear interpolation)

余裕のある人は、バイキュービック補間についてもおこなってください。加点があります。

# イメージモザイキング（image mosaicing）

- 少しずつ重なるようにして撮影された複数の画像を貼り合わせることで、1枚の広視野かつ高解像度の画像を合成する手法である。
- (1) 画像間の幾何学的変換パラメータの推定
  (2) 画像の幾何学的変換と色補正

■図9.20——イメージモザイキングの例（提供：三洋電機株式会社デジタルシステム技術開発センター）

# 幾何学的変換パラメータの推定

- **特徴点の自動対応付け**

  一方の画像から、物体のコーナーのような特徴点を抽出する。

  その特徴点に対応する点をもう一方の画像から探索し、画像間で対応する特徴点の座標の組を取得する

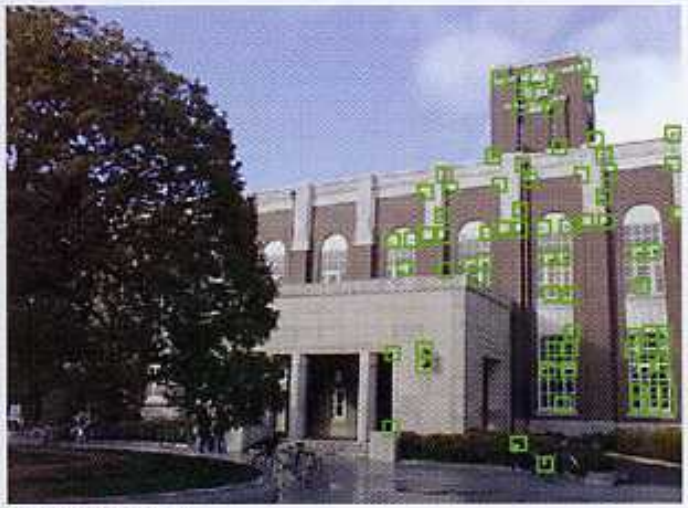
- **幾何学的変換パラメータの算出**

  対応付けられた特徴点の座標の組から隣接画像間の幾何学変換パラメータを算出する。

  アフィン変換の場合、未知のパラメータは6であり、3組の対応点が必要である。

  射影変換の場合、未知のパラメータは8であり、4組の対応点が必要である。

  それ以上の対応点が得られている場合、最小二乗法により誤差を軽減することができる。



[a] 特徴点抽出

[b] 対応付け

■図9.21——特徴点の抽出と対応付け（提供：三洋電機株式会社デジタルシステム技術開発センター）

# 画像の幾何学的変換と色補正

- 算出された基準画像に対する幾何学的変換パラメータを用いて、各画像の幾何学的変換を行い、基準画像につなぎ合わせていく。
- 画像間の明るさや色合いの違いによるつなぎ目をめたたなくするために、色補正を行う。補正法として、画像間の重なり領域の色の違いを、どちらの画像に近いかという距離を基準に補正する方法がある。

# 全周モザイキング

- カメラを三脚に固定し、水平方向に360度回転して画像を撮影することにより、全周モザイキングが可能である。
- 各画像をいったん円筒面に投影したあと、円筒面上において貼り合わせを行う。

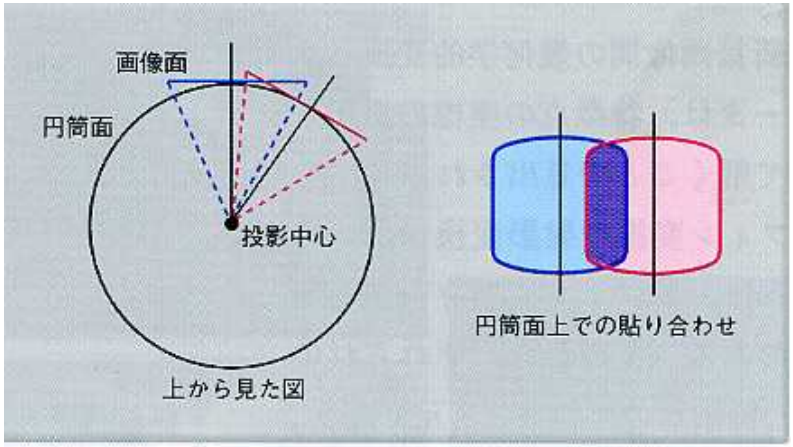



図9.22——円筒面上での貼り合わせ

図9.23——全周モザイキング（円筒パノラマ）の例（提供：三洋電機株式会社デジタルシステム技術開発センター）