アフィン変換

空間画像処理

- 画像の線形幾何学変換（拡大，縮小，回転）

- 座標 p(x,y)の位置の点が，変換によってp’(x’,y’)に移動するとき，次のように表現される

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} ax + by + e \\ cx + dy + f \end{bmatrix}$$

空間画像処理

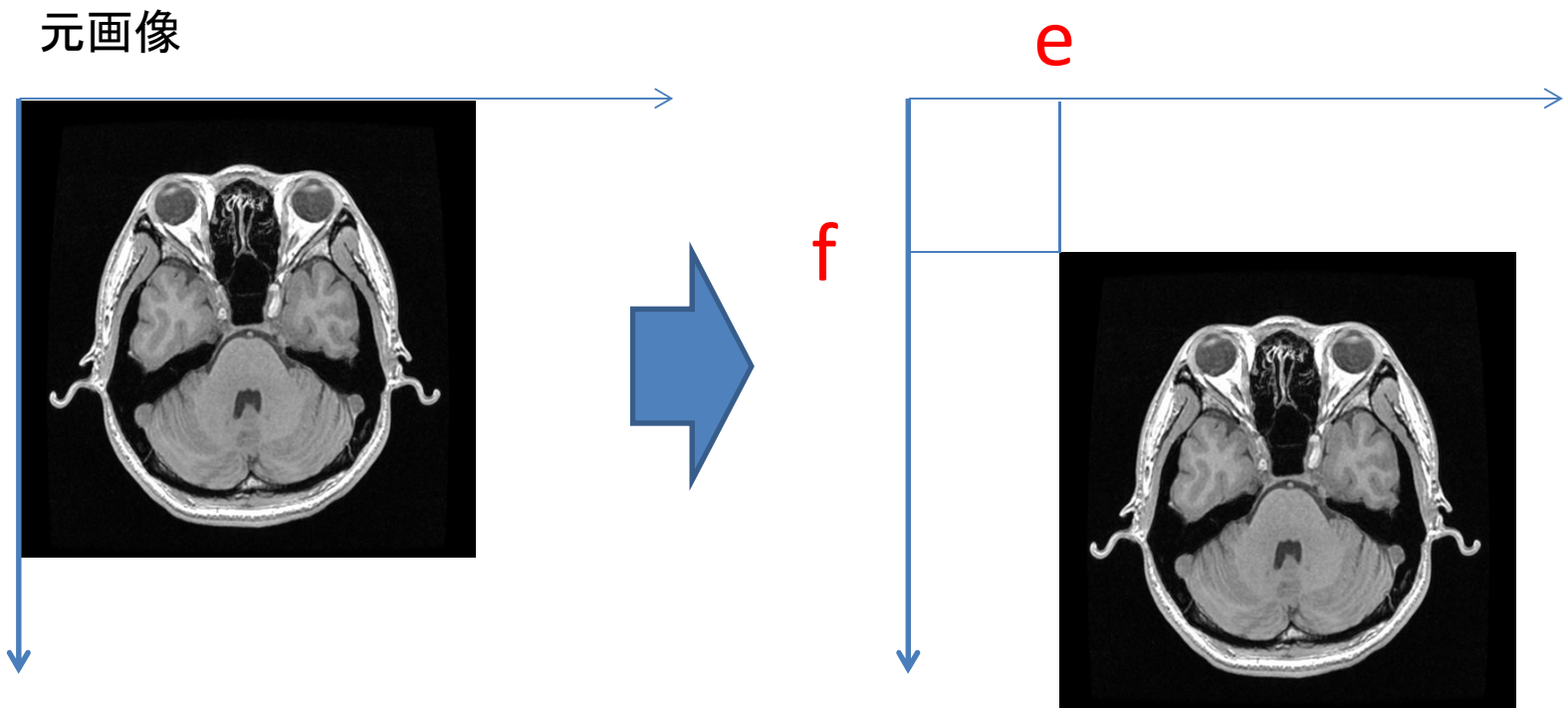
- 例えばa=d=1, c=b=0のとき

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} x + e \\ y + f \end{bmatrix}$$

となり，元の画像を平行移動した画像になる

空間画像処理

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} x + e \\ y + f \end{bmatrix}$$



課題

- x, y方向に平行移動し画像を出力するプログラムを作成せよ

- 指定した倍率で画像を拡大・縮小させるプログラムを作成せよ

- 指定した角度で画像を回転させるプログラムを作成せよ

演習１

- main関数の構造

```
load_image(0,"");
affine_shift(0,1);
save_image(1,"");
```

- 実行結果例

```
入力ファイル名 (*.pgm) : mri3.pgm
横の画素数 = 512, 縦の画素数 = 512
最大階調値 = 255
画像は正常に読み込まれました.
x方向の移動量を入力してください.
10
y方向の移動量を入力してください.
20
出力画像サイズ : 512, 512
移動しました
出力ファイル名 (*.pgm) : mri4.pgm
```

平行移動プログラム

```
void affine_shift(int n,int n2)
{
        int shift_x, shift_y;
        int size_x, size_y;
        int out_pos_x,out_pos_y;

        printf("x方向の移動量を入力してください．¥n");
        scanf("%d",&shift_x);

        printf("y方向の移動量を入力してください．¥n");
        scanf("%d",&shift_y);
```

```
        size_x = (int)(width[n]);
        size_y = (int)(height[n]);
        width[n2] = size_x;
        height[n2] = size_y;
        printf("出力画像サイズ:%d, %d¥n", size_x,size_y);
        for(int j=0;j<size_y;j++){
                for(int i=0;i<size_x;i++){
                        out_pos_y = j-shift_y;
                        out_pos_x = i-shift_x;
                        if(out_pos_x >= 0 && out_pos_x < width[n]
                            && out_pos_y >= 0 && out_pos_y < height[n]){
                        image[n2][i][j] = image[n][out_pos_x][out_pos_y];
                        }
                }
        }
        printf("移動しました");
}
```

出力画像の各画素のループ

元画像のどこの画素か

画素値を代入する

空間画像処理

- 例えば b=c=e=f=0のとき

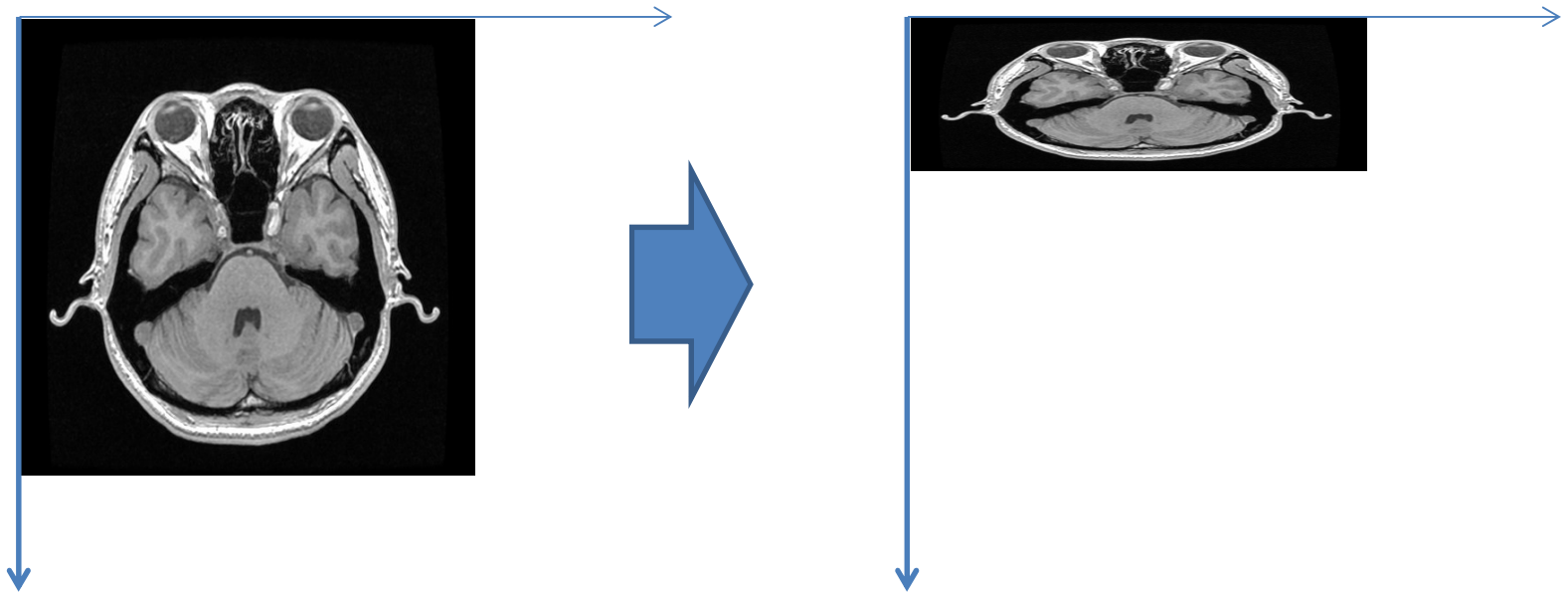
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ dy \end{bmatrix}$$

となり，拡大，縮小を表す．a, dはx，y方向の拡大（縮小）率を表す

空間画像処理

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ by \end{bmatrix}$$

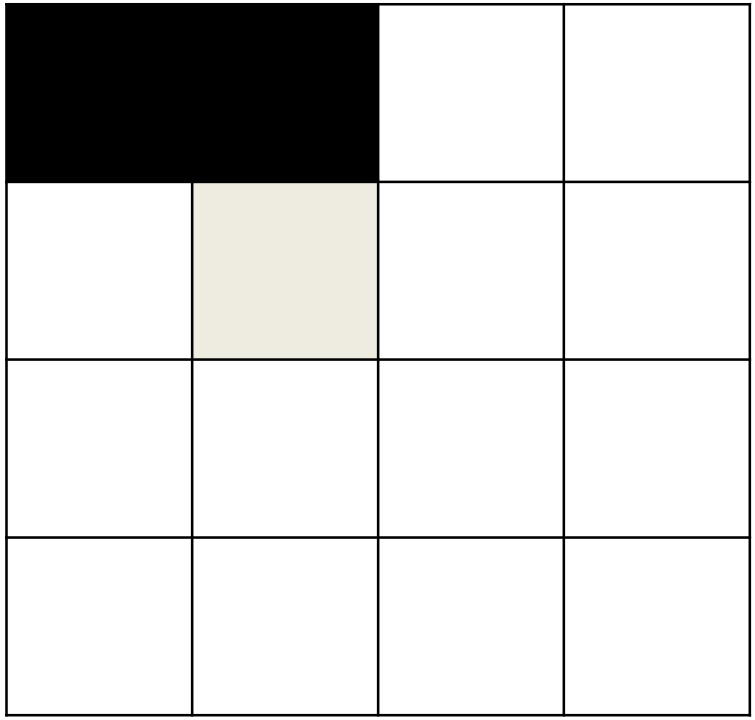
元画像



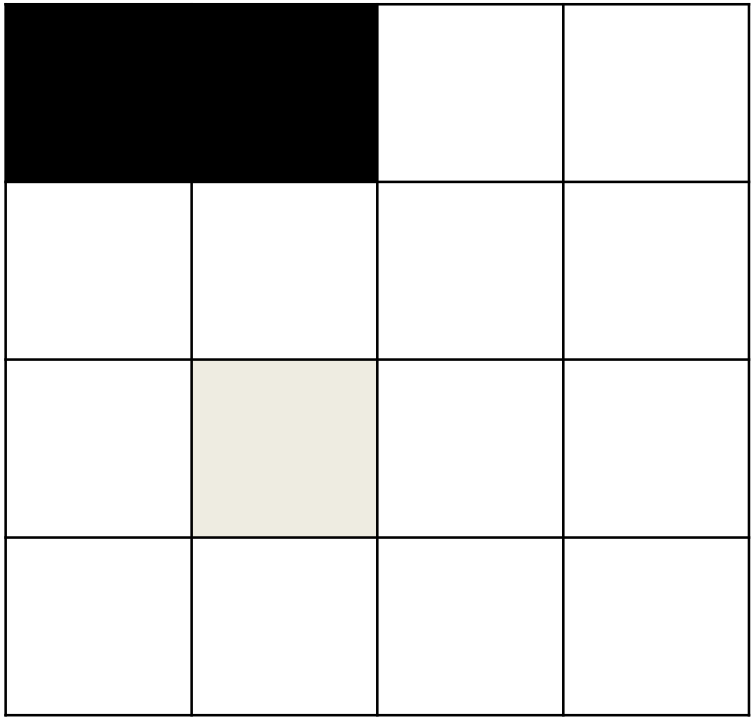
原画像の拡大･縮小

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ dy \end{bmatrix}$$

例えば a = 1,　d=2



x-y

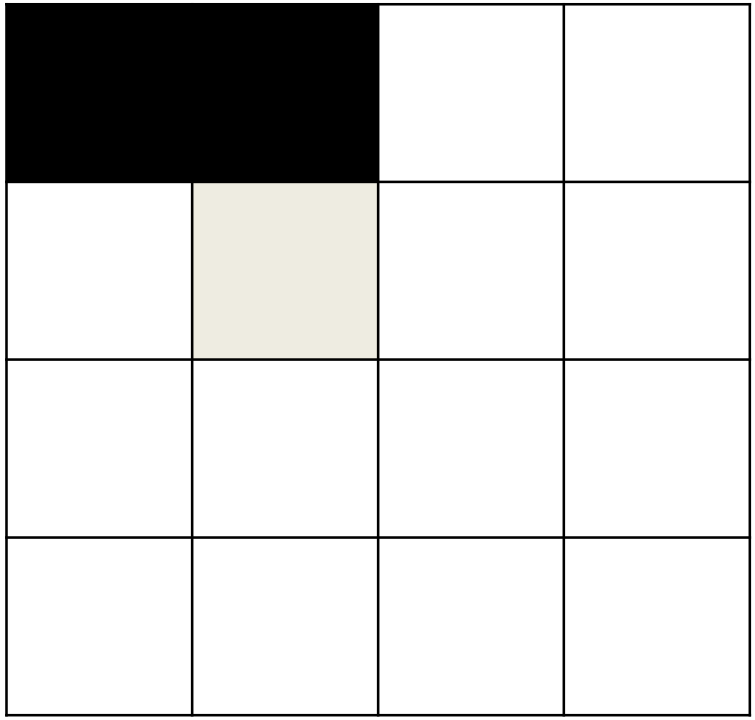


x’-y’

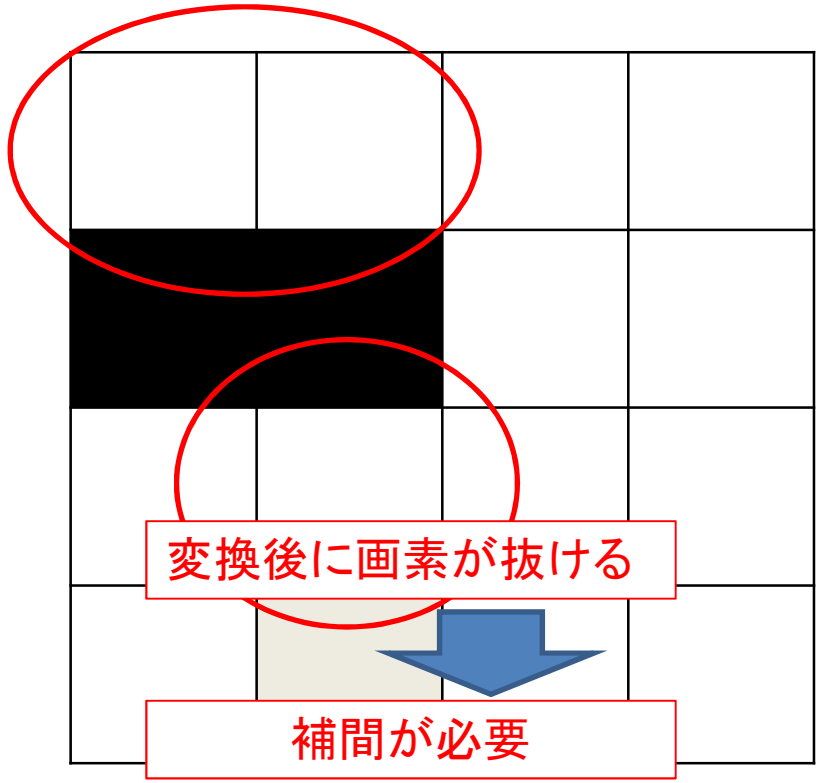
原画像の拡大･縮小

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} ax \\ dy \end{bmatrix}$$

例えば a = 1,　d=2



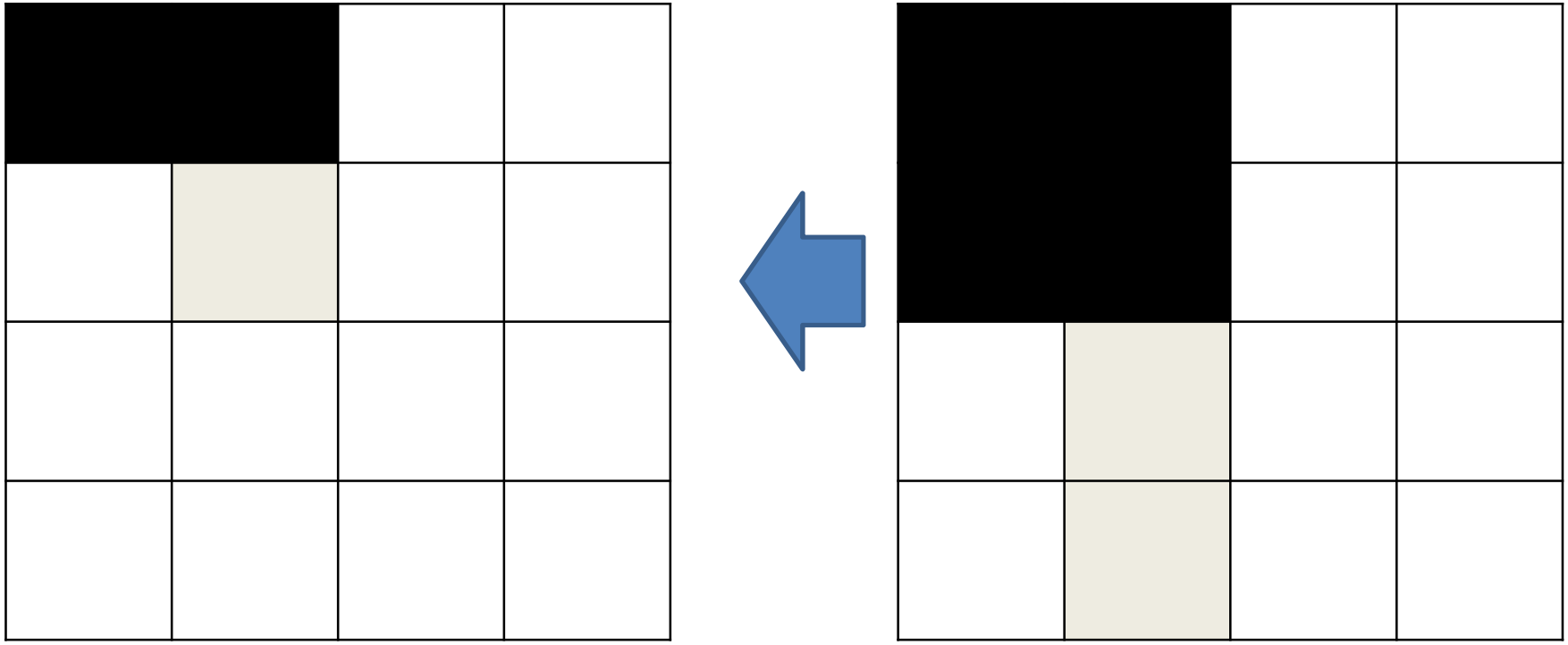
x-y



x’-y’

原画像の拡大･縮小

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x'/a \\ y'/d \end{bmatrix}$$

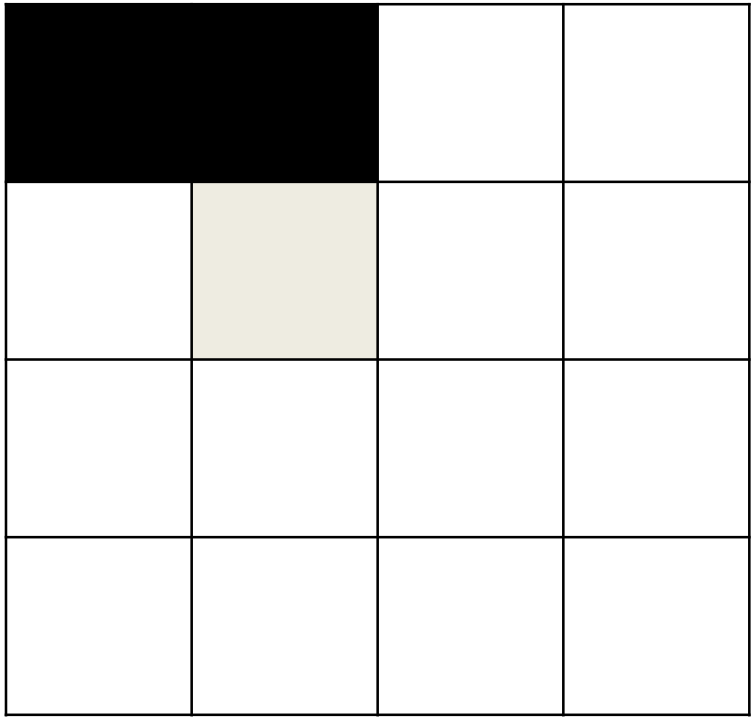


x-y

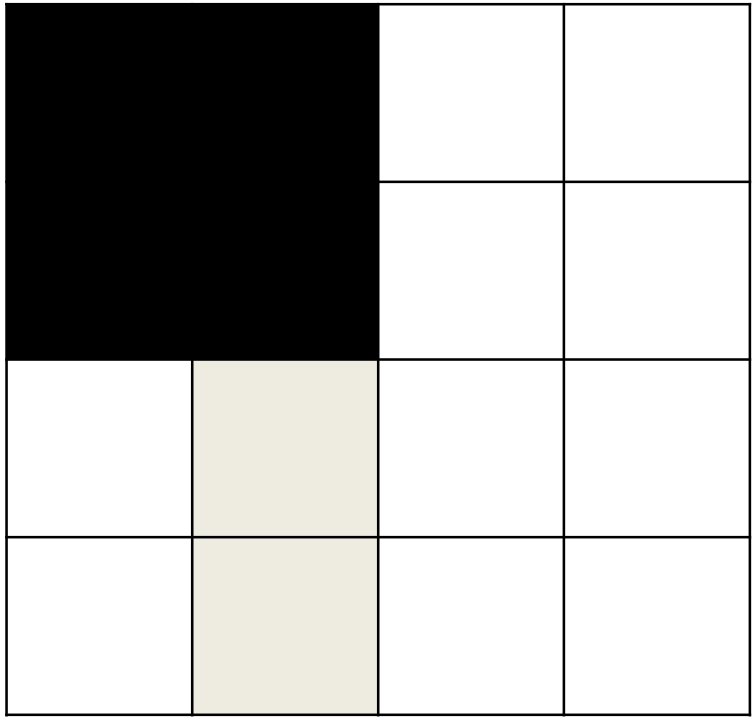
x’-y’

最近傍法(nearest neighbor)による補間

- 変換後の画素(x’, y’)
- = 元の画素((int)(x+0.5), (int)(y+0.5))



x-y



x’-y’

空間画像処理

- 例えば $a^2+b^2=c^2+d^2=1$のとき

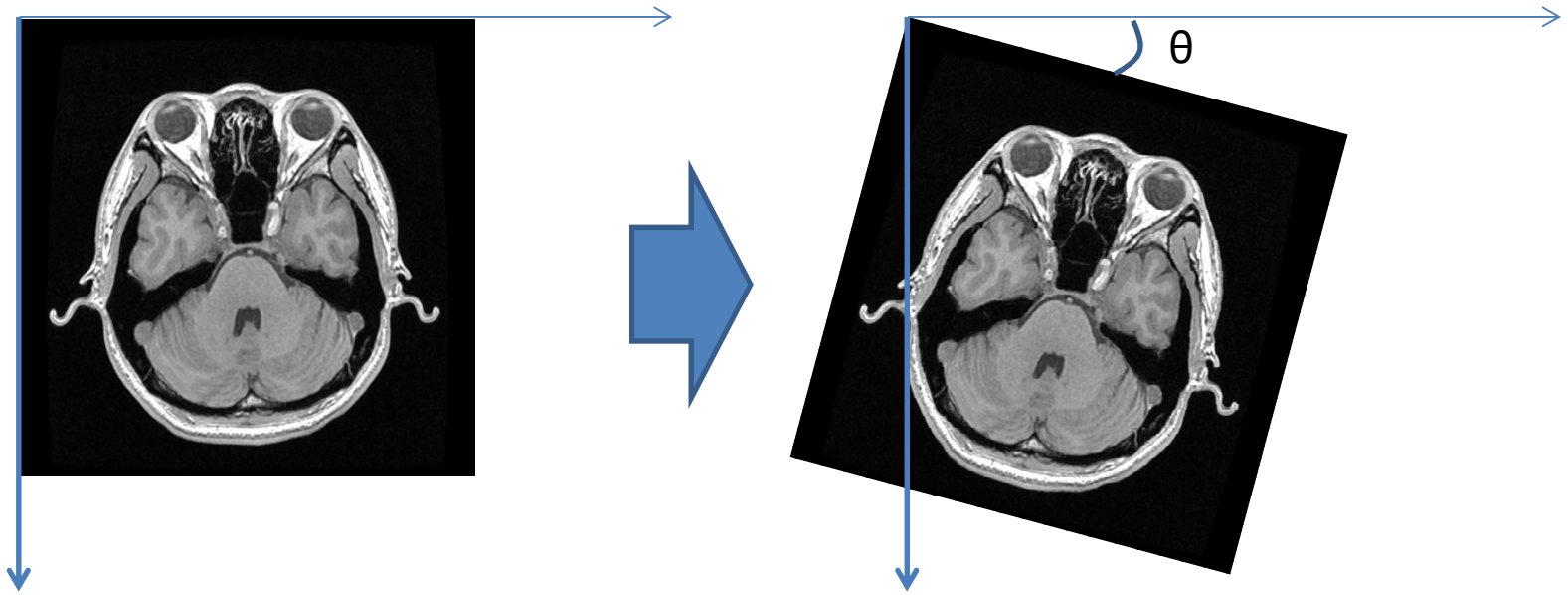
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} xcos\theta - ysin\theta \\ xsin\theta + ysin\theta \end{bmatrix}$$

となり，原点を中心とした角度θの回転を表す

空間画像処理

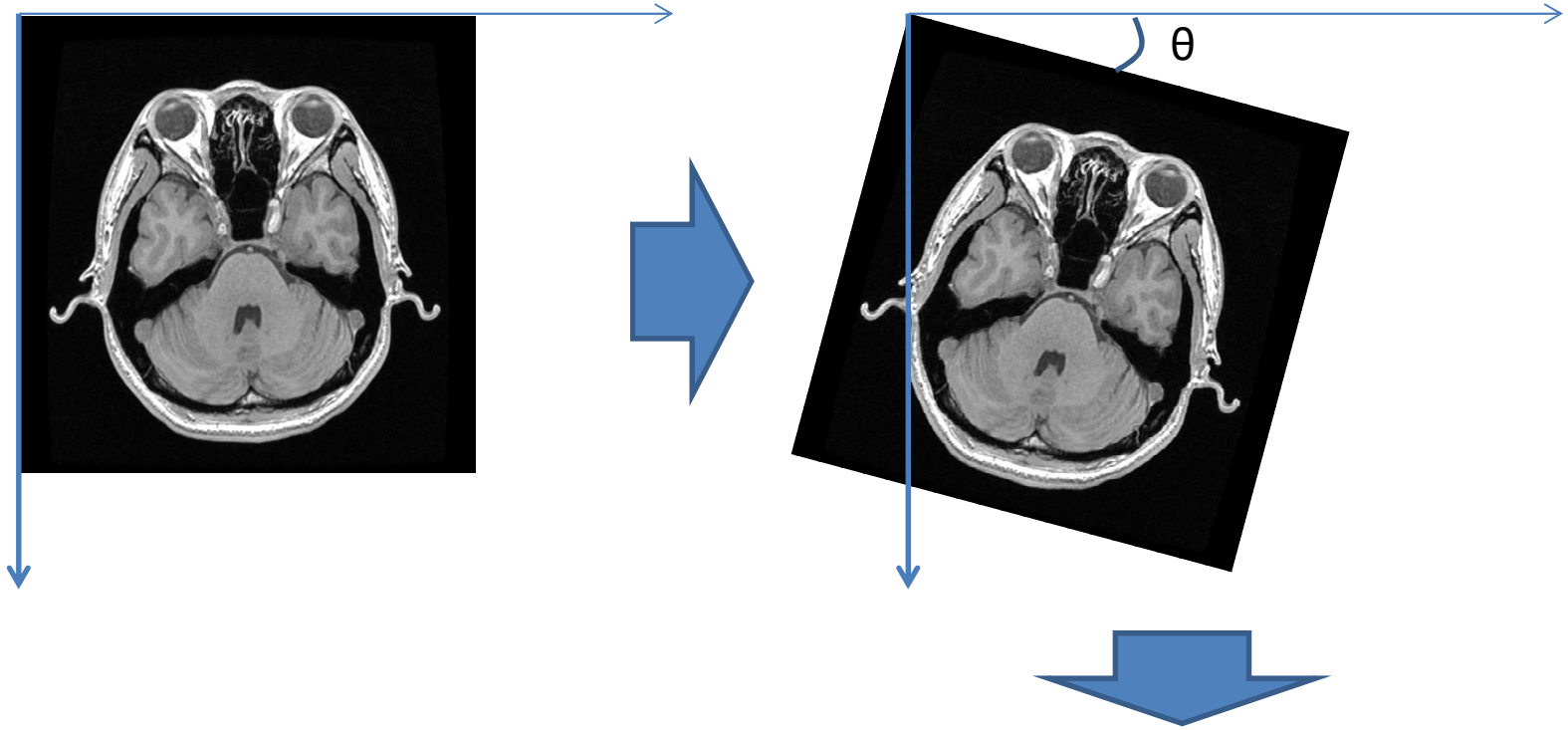
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} xcos\theta - ysin\theta \\ xsin\theta + ysin\theta \end{bmatrix}$$

元画像



画像の回転

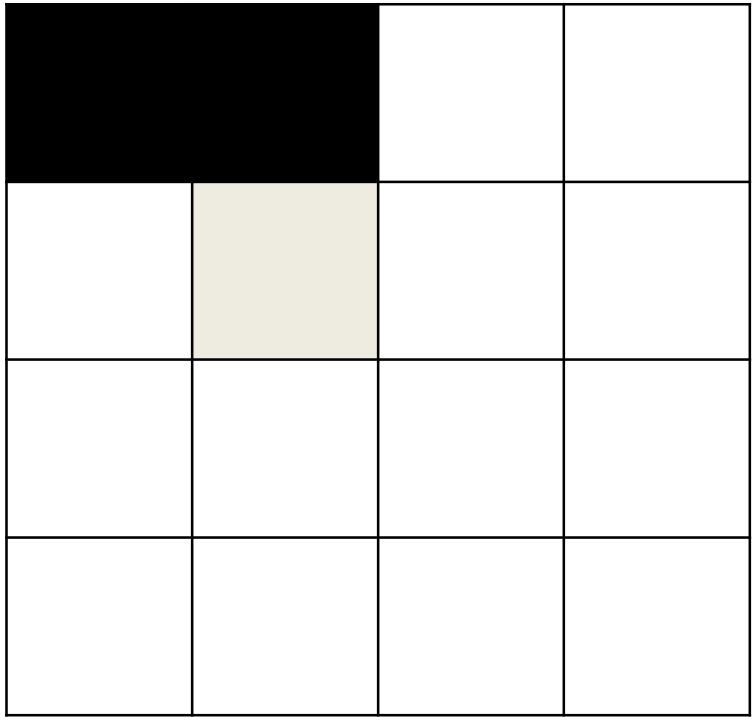
元画像



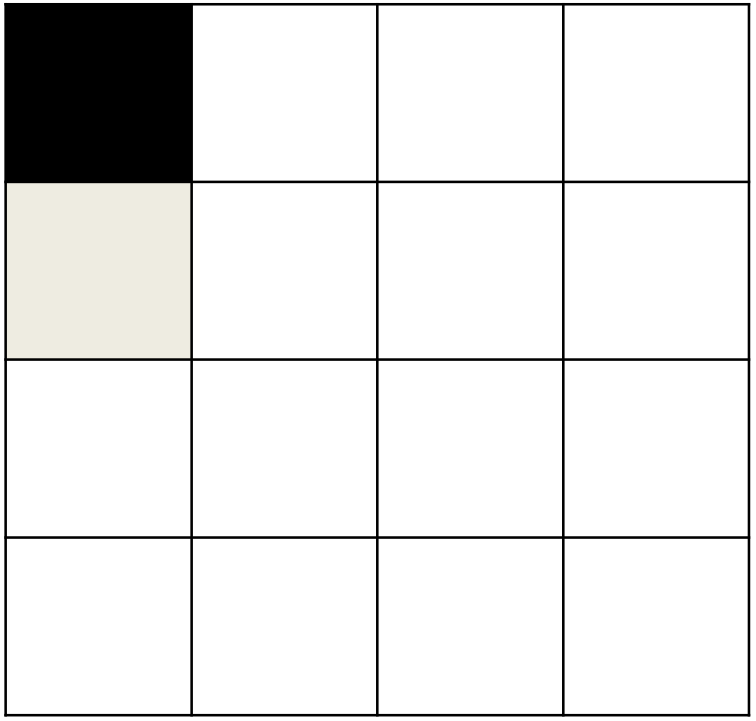
回転後の座標は実数であるため，
すべての画素が埋まるとは限らない

原画像を回転

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} xcos\theta - ysin\theta \\ xsin\theta + ysin\theta \end{bmatrix}$$



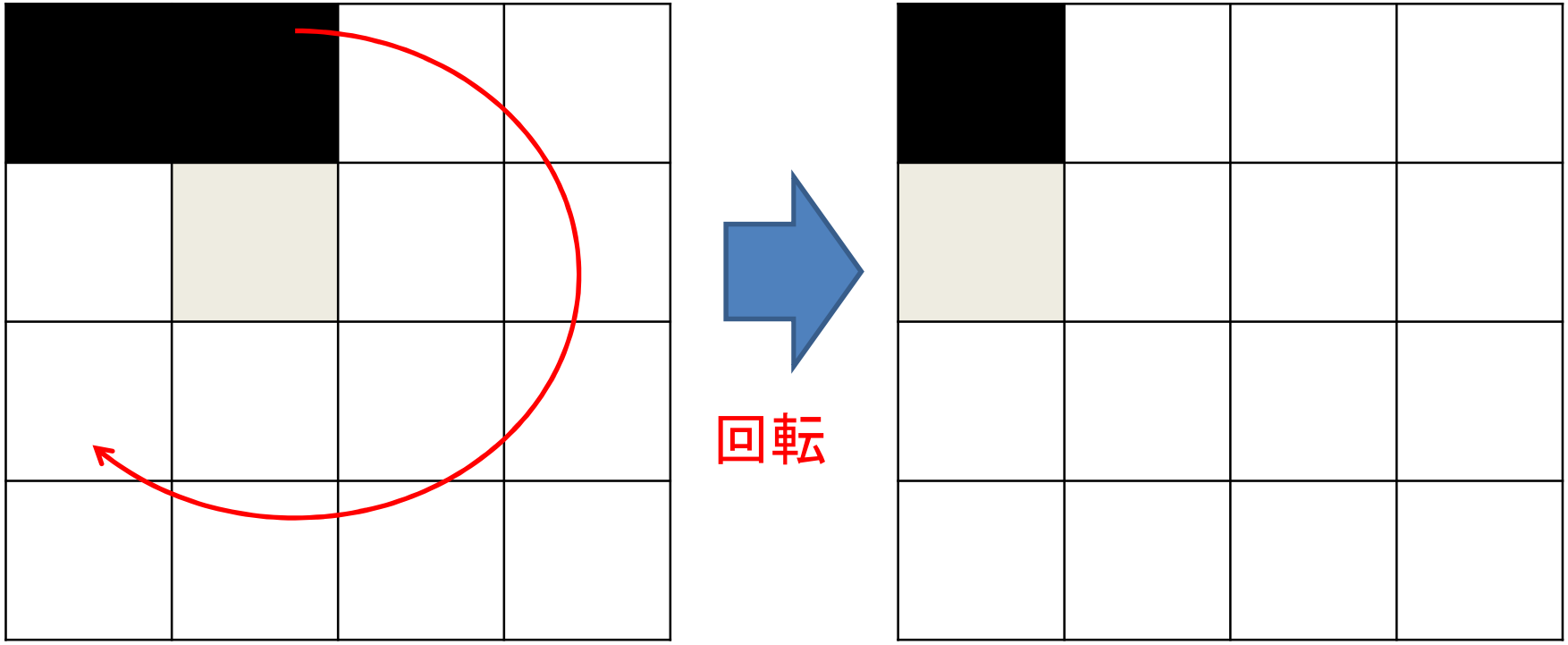
x-y



x’-y’

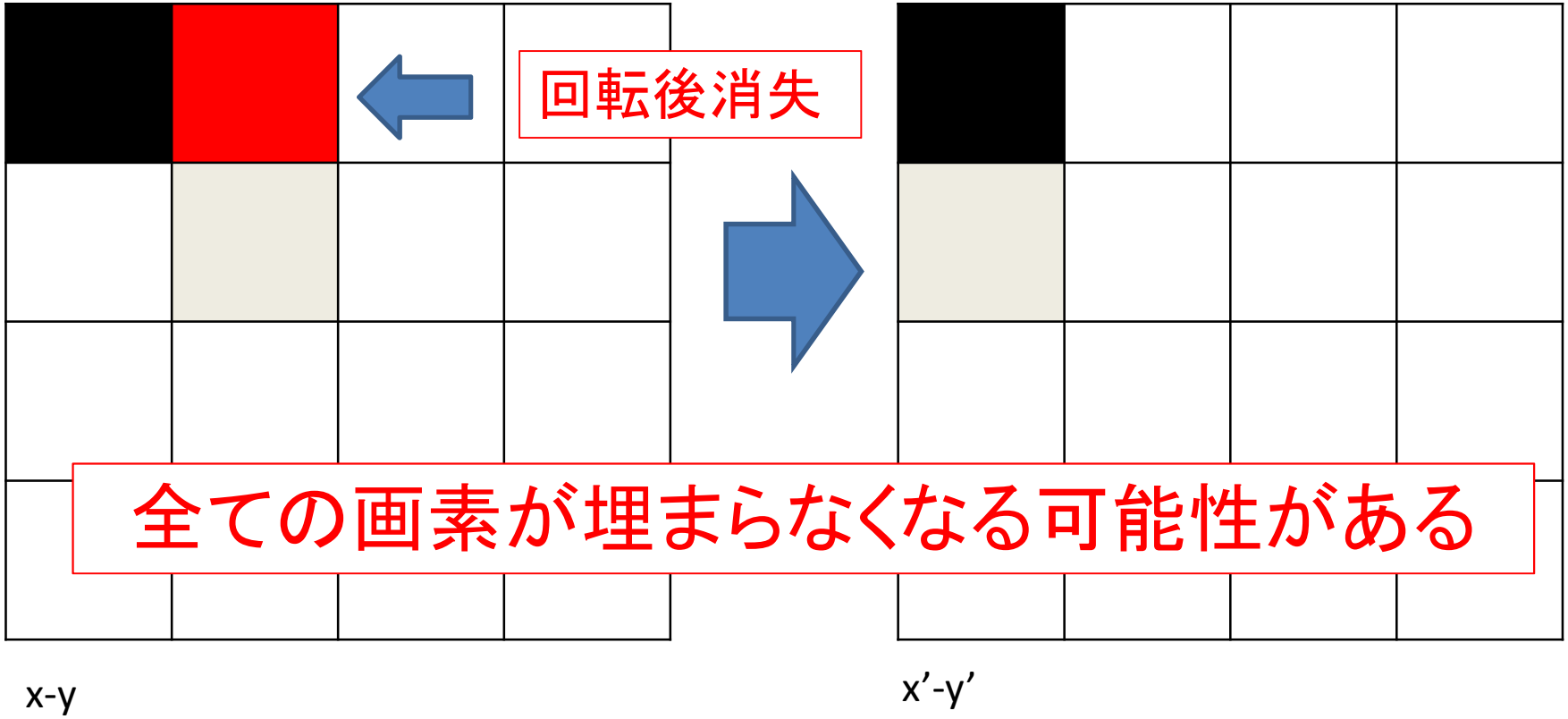
原画像を回転

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} xcos\theta - ysin\theta \\ xsin\theta + ysin\theta \end{bmatrix}$$



原画像を回転

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} xcos\theta - ysin\theta \\ xsin\theta + ysin\theta \end{bmatrix}$$



最近傍法を利用するために

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

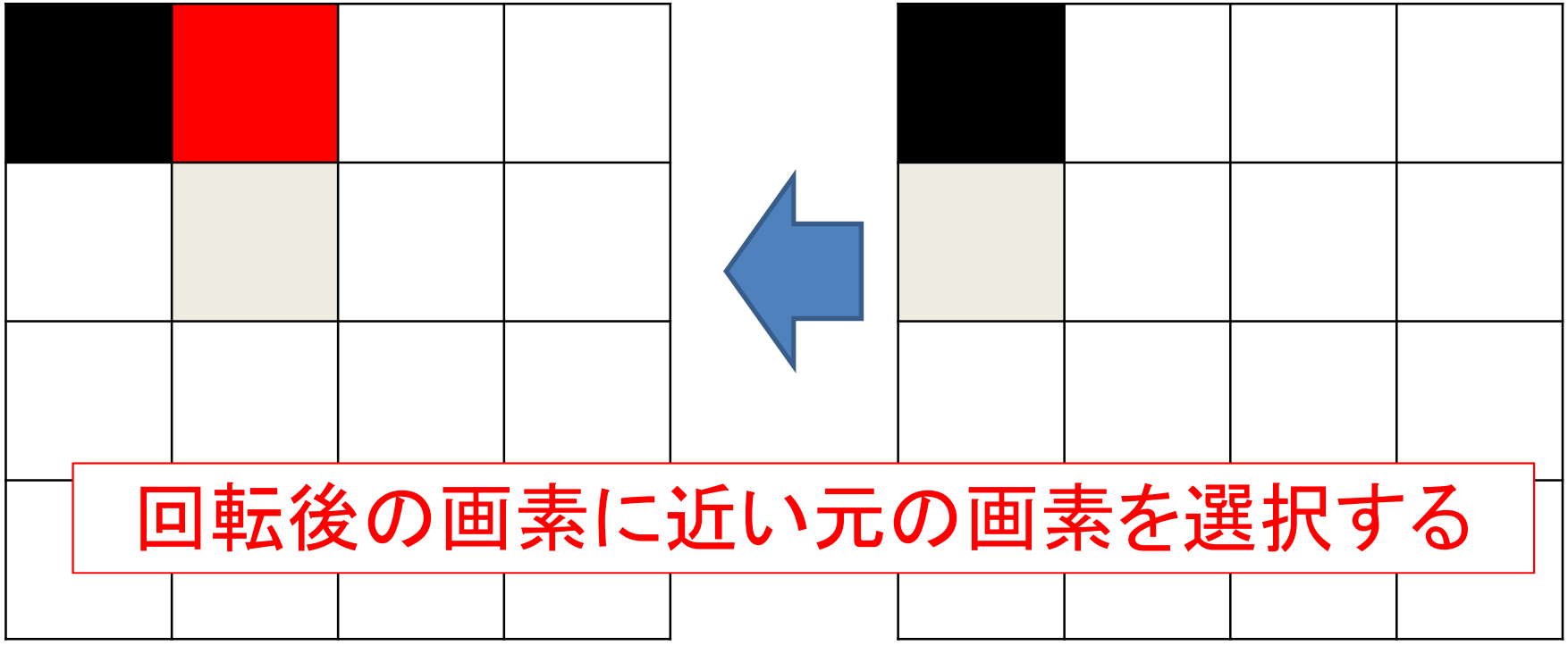
回転行列の逆行列を左からかける

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

全ての(x', y')に対して最近傍法による補間を行う

最近傍法

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



回転後の画素に近い元の画素を選択する

x-y

x’-y’

課題

- x, y方向に平行移動し画像を出力するプログラムを作成せよ

  affine_shift関数をコメントアウトして次に進む

- 指定した倍率で画像を拡大・縮小させるプログラムを作成せよ

- 指定した角度で画像を回転させるプログラムを作成せよ

演習１

- main関数の構造

```
load_image(0,"");
affine_size(0,1);
save_image(1,"");
```

- 実行結果例

```
入力ファイル名 (*.pgm) : mri3.pgm
横の画素数 = 512, 縦の画素数 = 512
最大階調値 = 255
画像は正常に読み込まれました.
x方向の拡大/縮小率を入力してください.
1.2
y方向の拡大/縮小率を入力してください.
2
出力画像サイズ : 614, 1024
サイズを変更しました
出力ファイル名 (*.pgm) : mri4.pgm
```

プログラムの雛型

```
void affine_size(int n,int n2)
{
        double ratio_x, ratio_y;
        int size_x, size_y;
        double pos_x,pos_y;
        int out_pos_x,out_pos_y;

        printf("x方向の拡大/縮小率を入力してください．¥n");
        scanf("%lf",&ratio_x);

        printf("y方向の拡大/縮小率を入力してください．¥n");
        scanf("%lf",&ratio_y);
```

```
        size_x = (int)(ratio_x*(double)width[n]+0.5);
        size_y = (int)(ratio_y*(double)height[n]+0.5);
        width[n2] = size_x;
        height[n2] = size_y;

        printf("出力画像サイズ:%d, %d¥n", size_x,size_y);

        for(int j=0;j<size_y;j++){
                for(int i=0;i<size_x;i++){
                        //この部分を作成する
                }
        }
        printf("サイズを変更しました");
}
```

課題

- x, y方向に平行移動し画像を出力するプログラムを作成せよ

- 指定した倍率で画像を拡大・縮小させるプログラムを作成せよ

  affine_size関数をコメントアウトして次に進む

- 指定した角度で画像を回転させるプログラムを作成せよ

演習1

- main関数の構造

```
load_image(0,"");
affine_rot(0,1);
save_image(1,"");
```

- 実行結果例

```
入力ファイル名 (*.pgm) : mri3.pgm
横の画素数 = 512, 縦の画素数 = 512
最大階調値 = 255
画像は正常に読み込まれました.
回転角度を入力してください.
30
出力画像サイズ : 512, 512
回転しました出力ファイル名 (*.pgm) : mri4.pgm
```

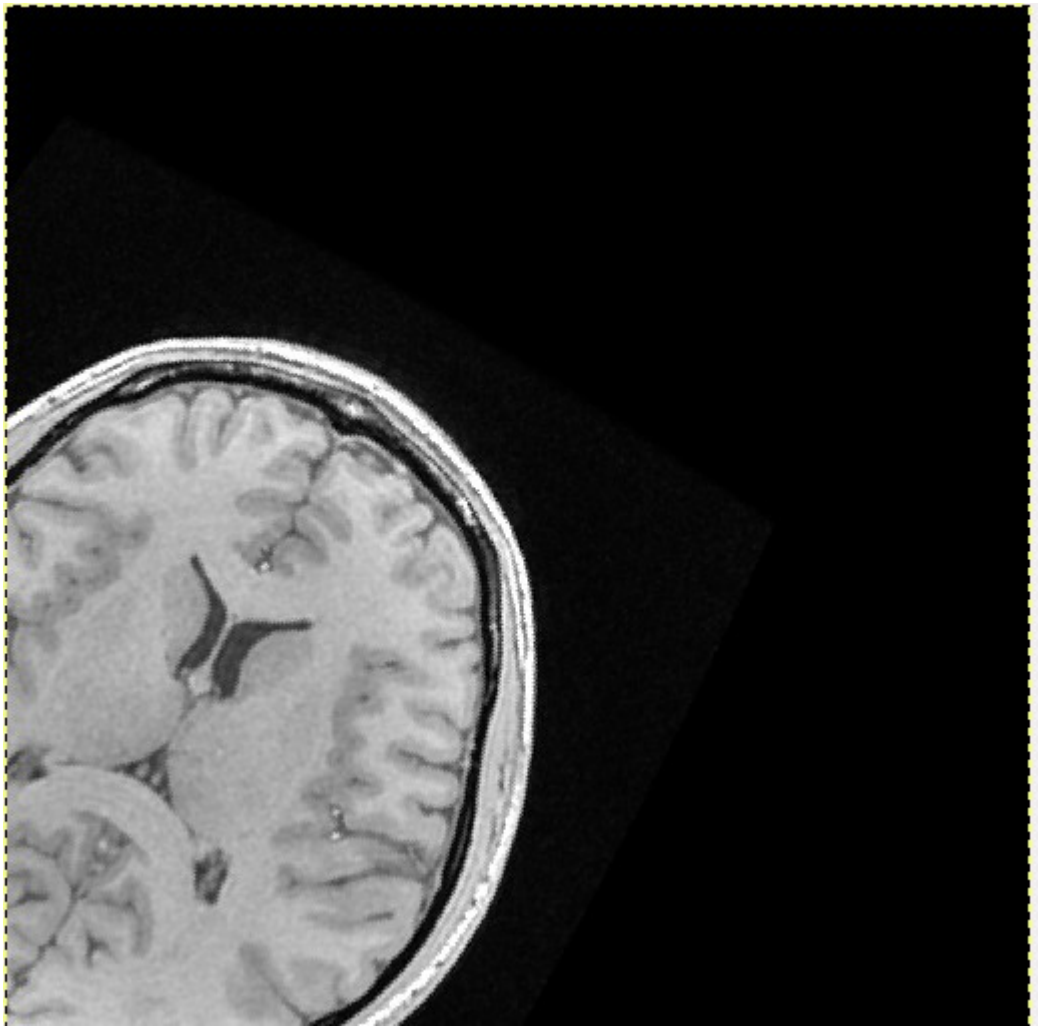
プログラムの雛型

```
void affine_rot(int n,int n2)
{
        double ratio_x, ratio_y;
        int size_x, size_y;
        double deg;
        double pos_x, pos_y;
        int out_pos_x,out_pos_y;

        printf("回転角度を入力してください．¥n");
        scanf("%lf",&deg);
```

```
        size_x = (int)(width[n]);
        size_y = (int)(height[n]);
        width[n2] = size_x;
        height[n2] = size_y;
        printf("出力画像サイズ:%d, %d¥n", size_x,size_y);
        for(int j=0;j<size_y;j++){
                for(int i=0;i<size_x;i++){
                        pos_x= (double)(i*cos(deg*PI/180.0))
                                        +(double)(j*sin(deg*PI/180.0));
                        pos_y = -(double)(i*sin(deg*PI/180.0))
                                        +(double)(j*cos(deg*PI/180.0));
                        //ここに作成すること
                }
        }
        printf("回転しました");
}
```

実行結果画像



余力があれば画像の中心を原点にして回転させること