Example: Image Rotation

- In a continuous signal, the rotation can be done using the following equation.

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

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

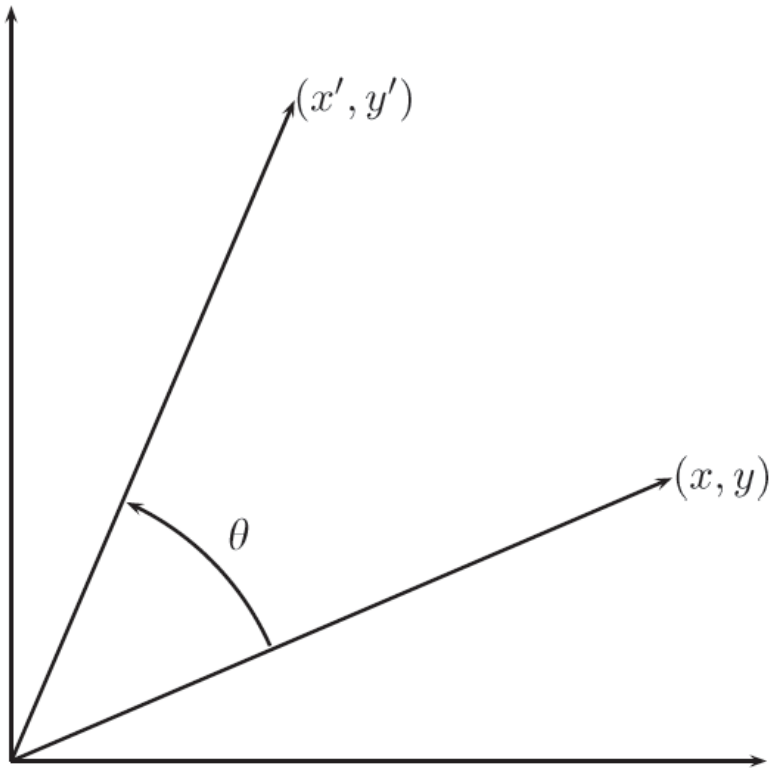


FIGURE 6.20 *Rotating a point through angle θ.*

Example: Image Rotation

- But, in a discrete signal?
  - The rotation does NOT always produce an integer pixel location.
  - So, the **interpolation** (or **re-sampling**) is needed.

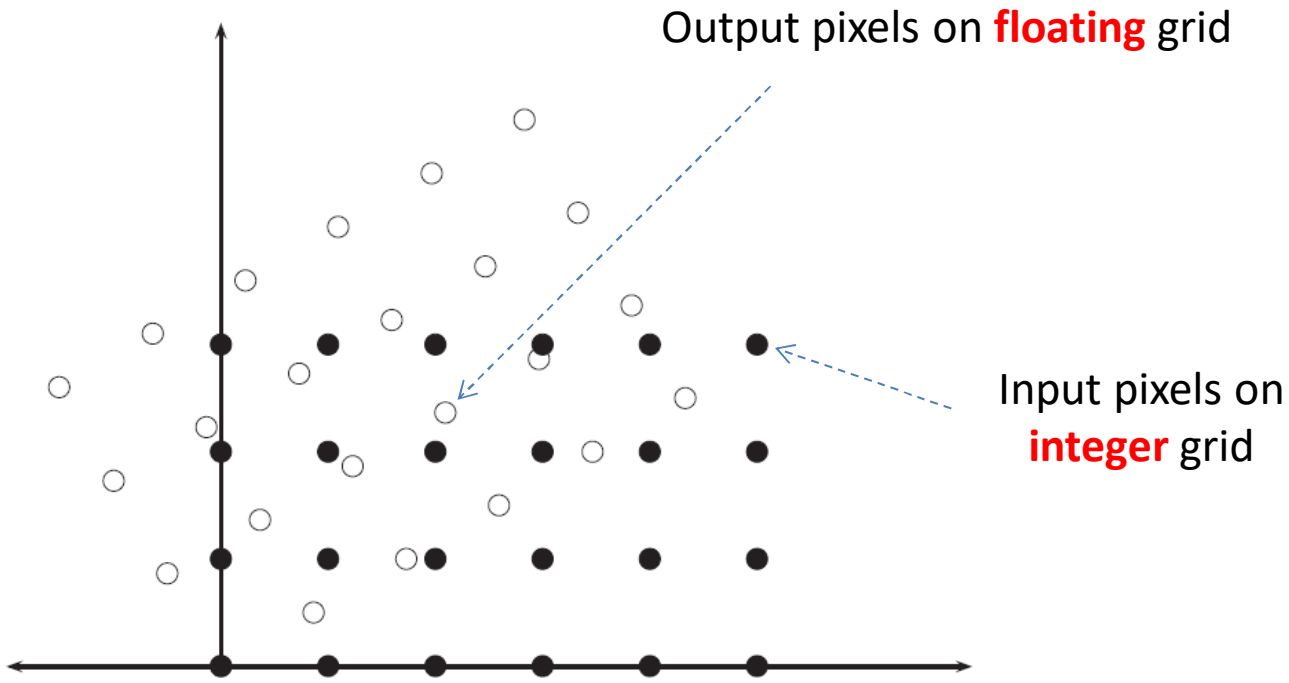


FIGURE 6.21 *Rotating a rectangle.*

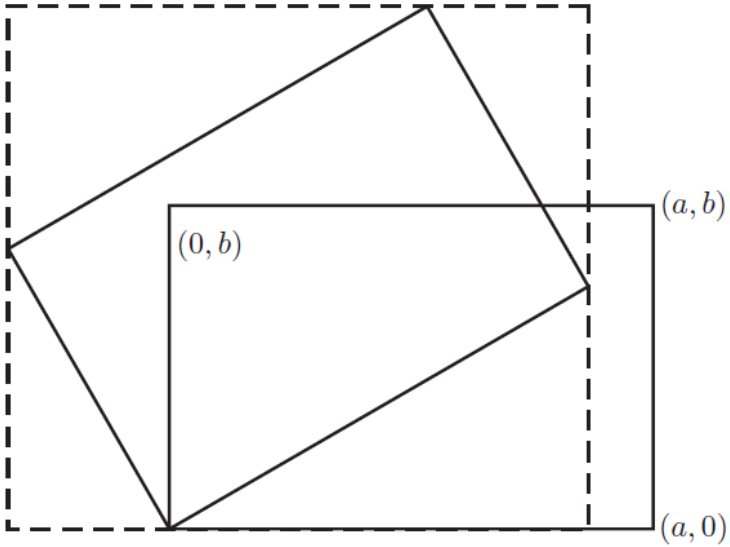


FIGURE 6.22 *A rectangle surrounding a rotated image.*

After rotation, an intensity value on a nearby integer grid is computed using the interpolation technique.

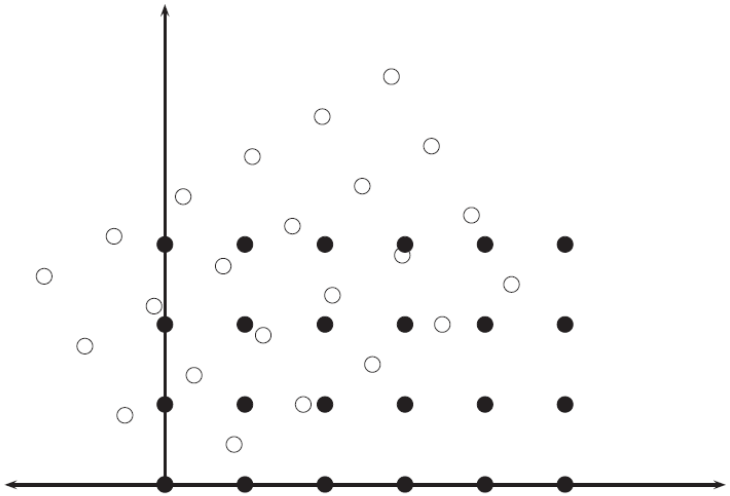


FIGURE 6.21 *Rotating a rectangle.*

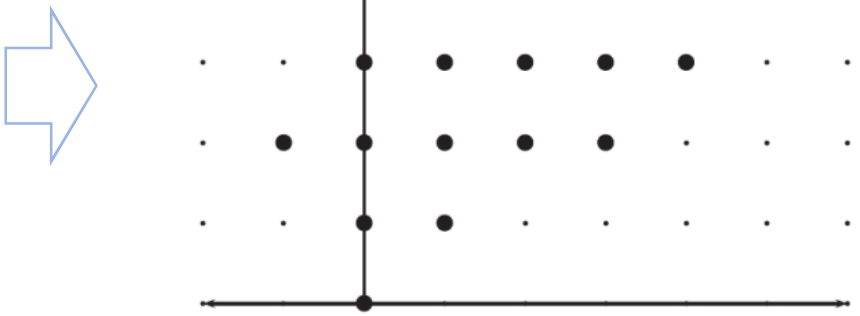


FIGURE 6.23 *The points on a grid after rotation.*

Example: Image Rotation

- Overall process

1. Define an enlarged, rotated image I_R whose size is larger than that of an original image. (ex. $2H \times 2W$)
2. Assume $(x_R, y_R) \in I_R$. Then, compute the transform $(x_R, y_R) \rightarrow (x', y')$. Note that (x', y') is a floating pixel.

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

3. Compute $f(x', y')$ using the interpolation technique.
4. Then, $f(x_R, y_R) = f(x', y')$

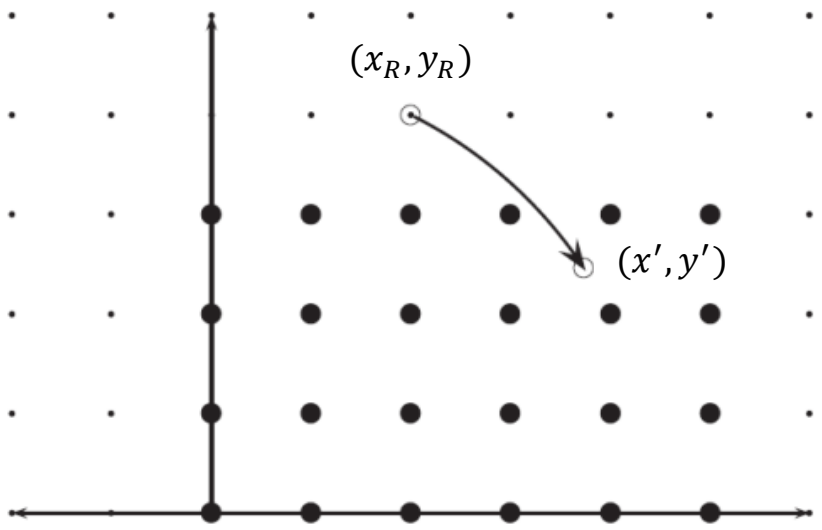


FIGURE 6.24 *Rotating a point back into the original image.*

Example: Image Rotation

Original image

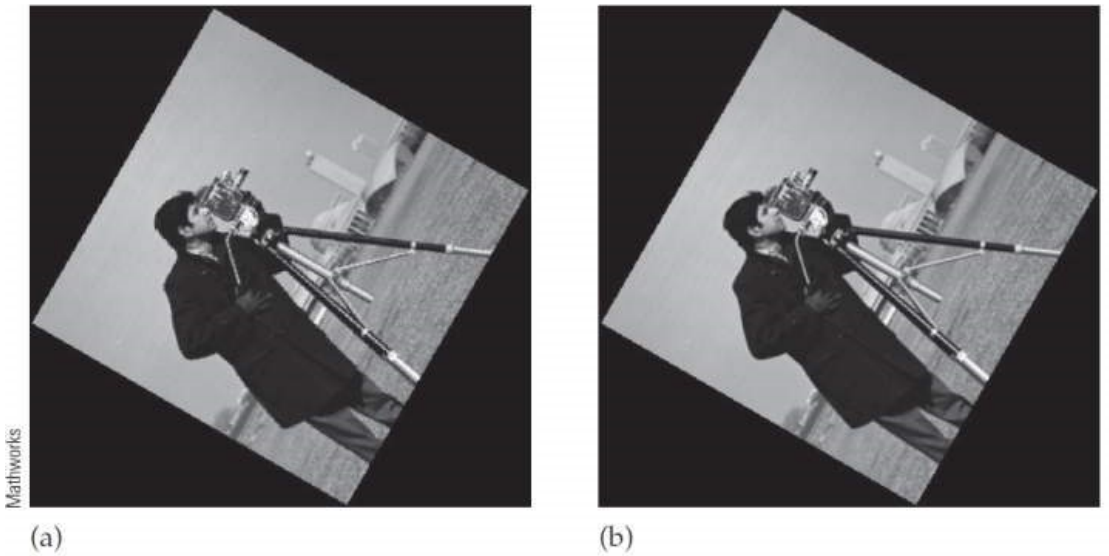
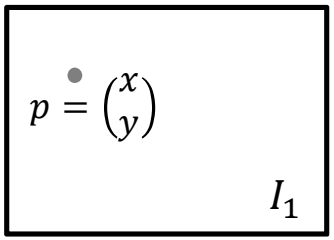
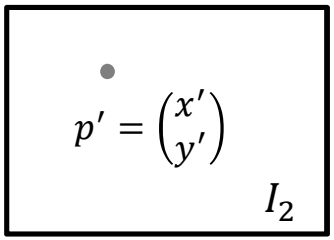


FIGURE 6.25 *Rotation with interpolation. (a) Nearest neighbor. (b) Bicubic interpolation.*

Example: Image Stitching using Affine Transformation

1. Estimate the affine transform





For a pair of corresponding pixels

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

For $N \geq 3$ pairs of corresponding pixels, affine transform for $I_1 \rightarrow I_2$ can be computed as follows.

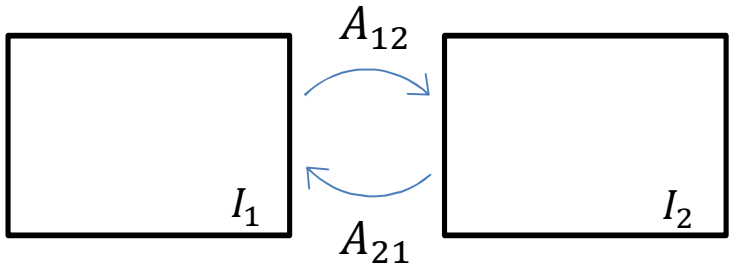
$$\boldsymbol{Mx} = \boldsymbol{b} \rightarrow \boldsymbol{x} = (\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M})^{-1}\boldsymbol{M}^{\mathrm{T}}\boldsymbol{b}$$

Similarly, affine transform for $I_2 \rightarrow I_1$ can be obtained.

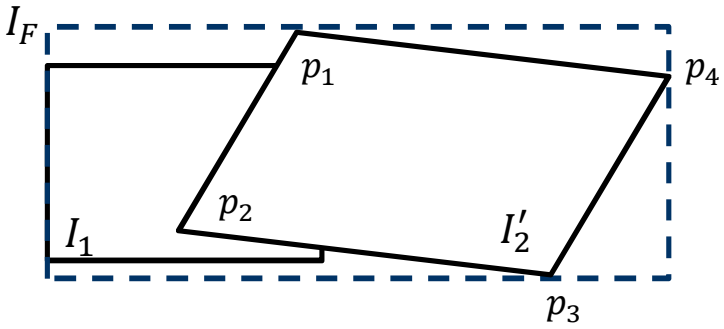
$$\underset{\boldsymbol{M}}{\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_N & y_N & 1 \end{pmatrix}} \underset{\boldsymbol{x}}{\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix}} = \underset{\boldsymbol{b}}{\begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ \vdots \\ x_N' \\ y_N' \end{pmatrix}}$$

Example: Image Stitching using Affine Transformation

2. Merge two images (I_2 to I_1)



2-1. The size of a final merged image I_F can be estimated by computing p_1, p_2, p_3, p_4 using A_{21}.



2-2. Perform the inverse warping using A_{12} within the region consisting of corners $[p_1, p_2, p_3, p_4]$.

2-3. Blend two images I_1 and I_2'.

$$I_F = \begin{cases} \alpha I_1 + (1-\alpha) I_2' & if\ both\ I_1\ and\ I_2\ are\ valid \\ I_1 & if\ only\ I_1\ is\ valid \\ I_2' & if\ only\ I_2'\ is\ valid \\ 0 & otherwise \end{cases}$$

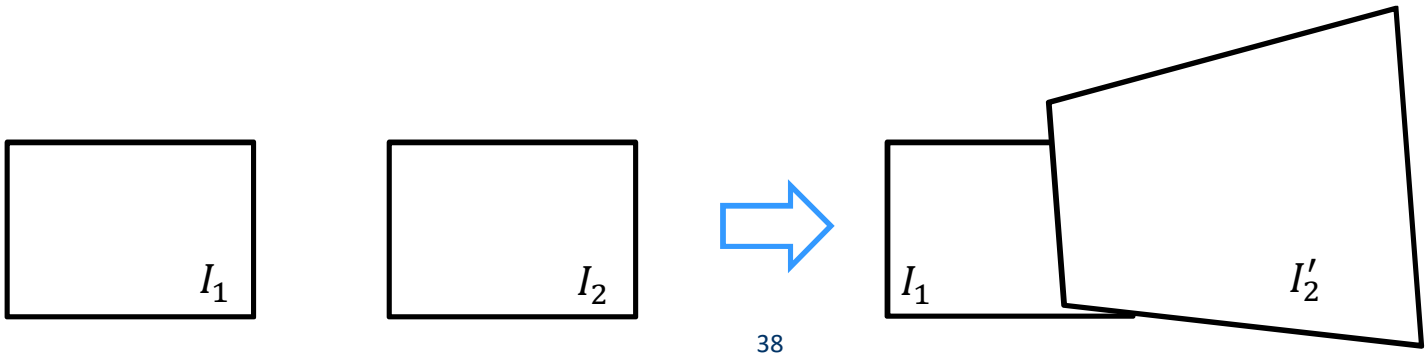
Example: Image Stitching using Projective Transformation

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

Projective Transformation

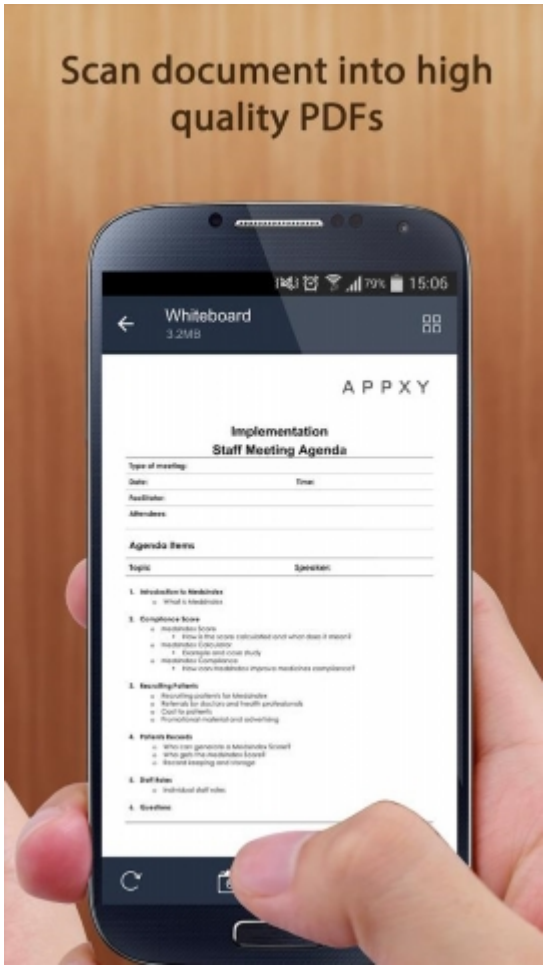
For $N \geq 4$ pairs of corresponding pixels

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 & -x_N'x_N & -x_N'y_N & -x_N' \\ 0 & 0 & 0 & x_N & y_N & 1 & -y_N'x_N & -y_N'y_N & -y_N' \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$



Example: Image Scanner using Projective Transform

Correcting project distortions



Summary

- Spatial resolution: image interpolation
  1. Compute a new pixel location
  2. Compute an intensity value for the pixel

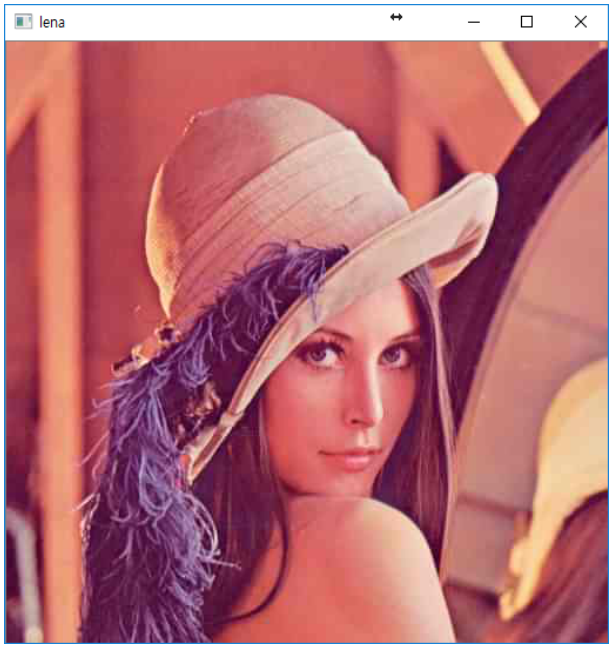
- Interpolation methods
  - Nearest-neighbor, Bilinear, Bicubic interpolation

- Image down-sampling
  - A pre-filtering is needed for anti-aliasing

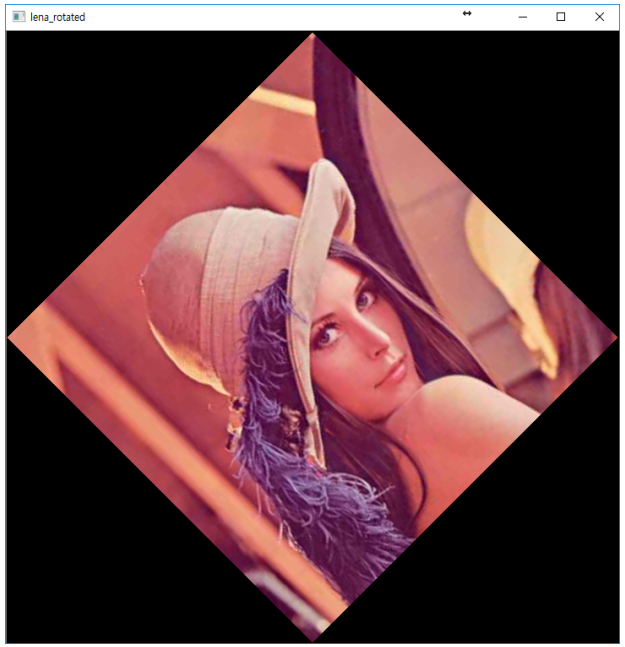
- Geometric Transformation
  - Translation, Rigid (Euclidean) transformation, Similarity transformation, Affine transformation, Projective transformation
  - Inverse warping is preferred.

Practice

- Image rotation
  - Implement the function below with the given skeleton code
  - Mat myrotate(const Mat input, float angle, const char* opt)
    - input : input matrix which will be rotated
    - angle : angle to rotate
    - opt : interpolation method
      - “nearest” or “bilinear”
      - implement both interpolation method



Rotate 45 degree



Practice

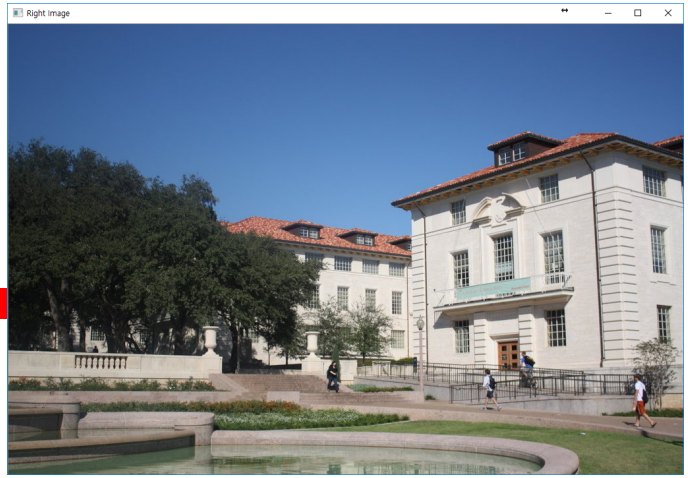
- Image Stitching using Affine Transformation
  - Estimate Affine transformation matrix
    - $Mx = b \rightarrow x = (M^T M)^{-1} M^T b$
  - Corresponding points are given below.
    Or, you can freely choose a set of corresponding points by yourself.
    - Image1_x : { 528, 505, 500, 482, 530, 645 };
    - Image1_y : { 509, 686, 768, 846, 917, 968 };
    - Image2_x : { 488, 469, 467, 452, 500, 609 };
    - Image2_y : { 45, 234, 314, 392, 456, 493 };
  - Implement both “forward warping” and “inverse warping” (p.32, p.33)
  - When stitching two images, use the “bilinear interpolation”
  - Refer to p.39 and 40 for more details.

Practice

- Image Stitching using Affine Transformation

image1

image2



Merged image