Chapter 5 Image Restoration

In this chapter, we focus on geometric transformations that modify the spatial relationships between pixels in an image.

Applications

- Image restoration (calibration, rectification)
- Image warping and morphing

Topics to be Covered

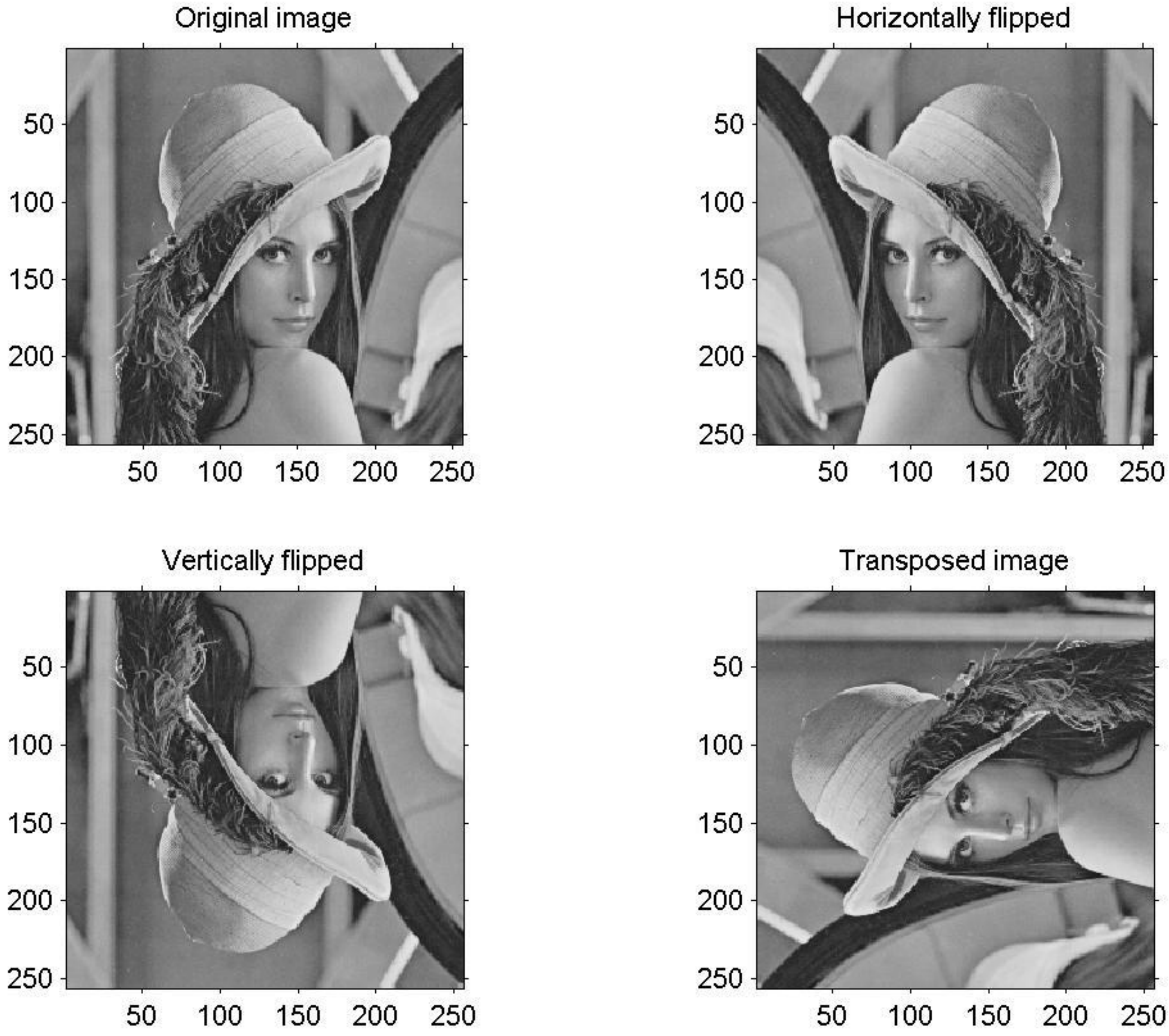
- Forward and reverse mapping
- Interpolation (NNI, Bilinear)
- Image translation
- Image rotation
- Image scaling (zooming)
- Affine transform
- Radial distortion
- Image warping

Simple Image Manipulations

A digital image is a matrix. Any manipulation of rows and columns results in an image manipulation.
(E.g.)

1. Horizontally flipped image (Mirror image)
   Rearrange all columns in the reverse order.
2. Vertically flipped image (Upside-down image)
   Rearrange all rows in the reverse order.
3. Transposed image
   Mirror followed by 90-degree anti-clockwise rotation
4. Image translation
   Shift the indexes of rows and/or columns

Examples of Image Manipulations



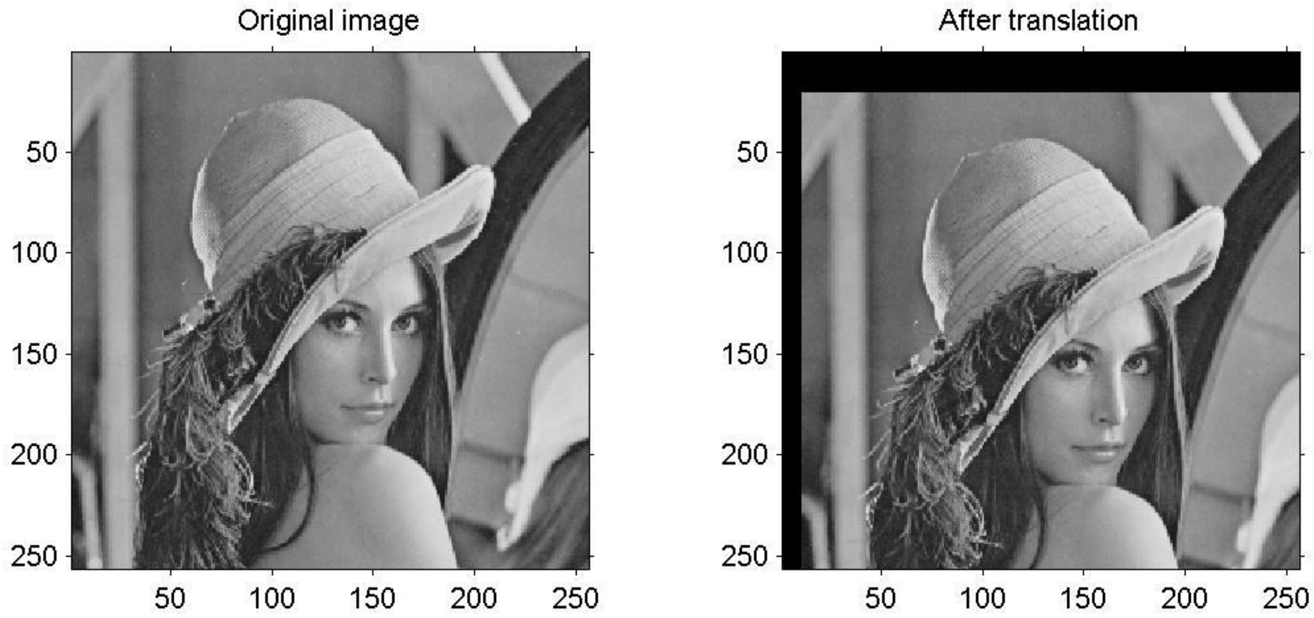
MATLAB Commands

```
>> img=imreas('Lena.gif');
>> [vsize,hsize]=size(img);
>>
>> mirror=img(1:vsize,hsize:-1:1);
>>
>> upside=img(vsize:-1:1,1:hsize);
>>
>> trans=transpose(img);
>>
```

Exercise 1

How do you generate a 180-degree rotated image by manipulating the columns and rows of the original image?

Image is shifted vertically by 20 pixels, and horizontally by 10 pixels as



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 20 \\ 10 \end{pmatrix}$$

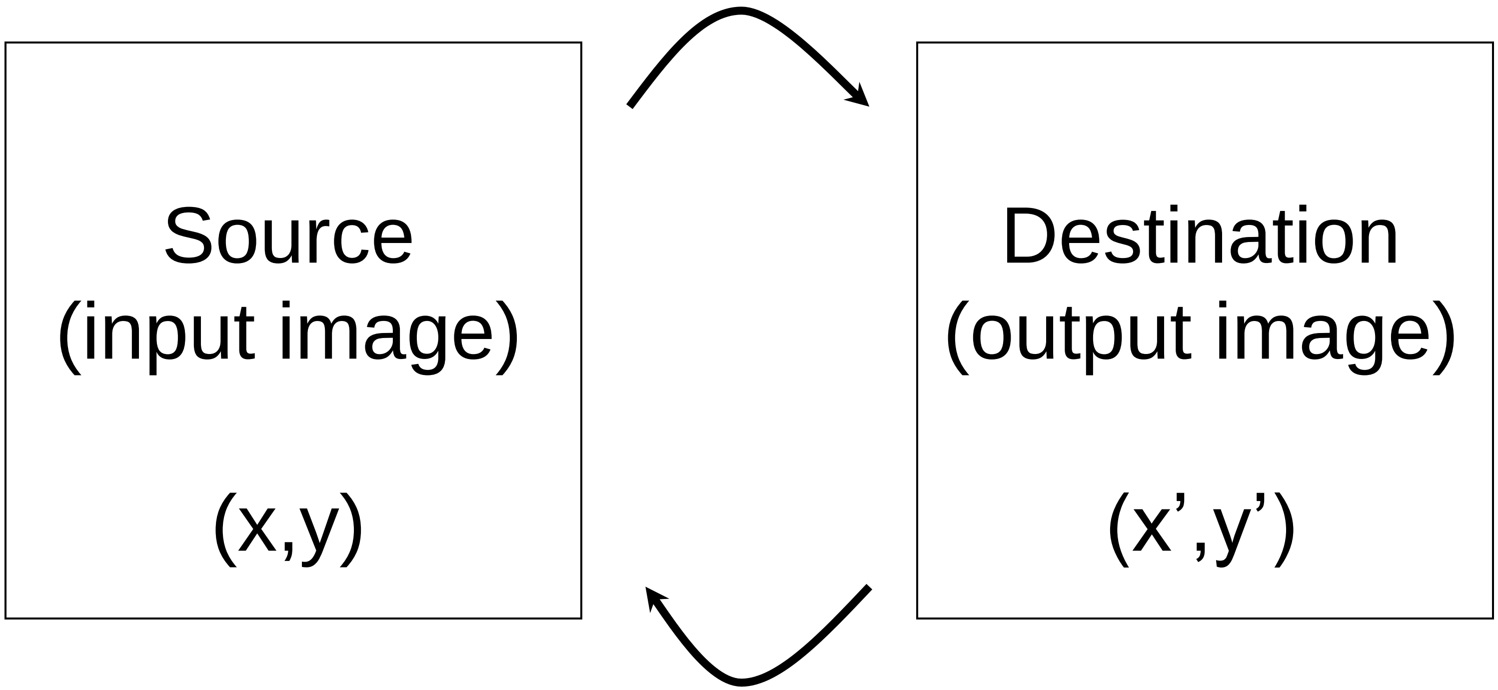
Forward and Reverse Mapping

There are two ways to generate a new image from an original image; forward mapping and reverse mapping.

In either case, we need to have equations that relate the coordinates in the original image and the new image.

Forward and Reverse Mapping

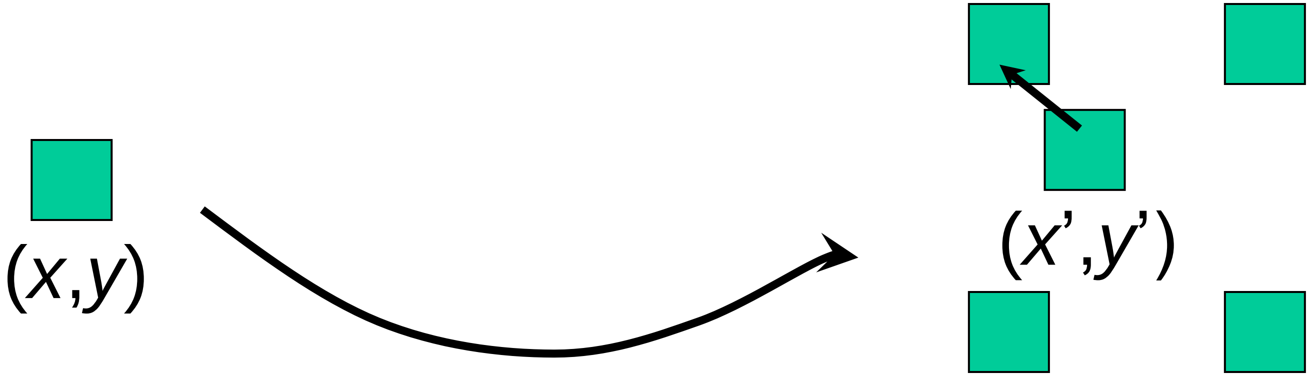
Forward mapping
(Read address control)



Reverse mapping or backward mapping
(Write address control)

Forward Mapping

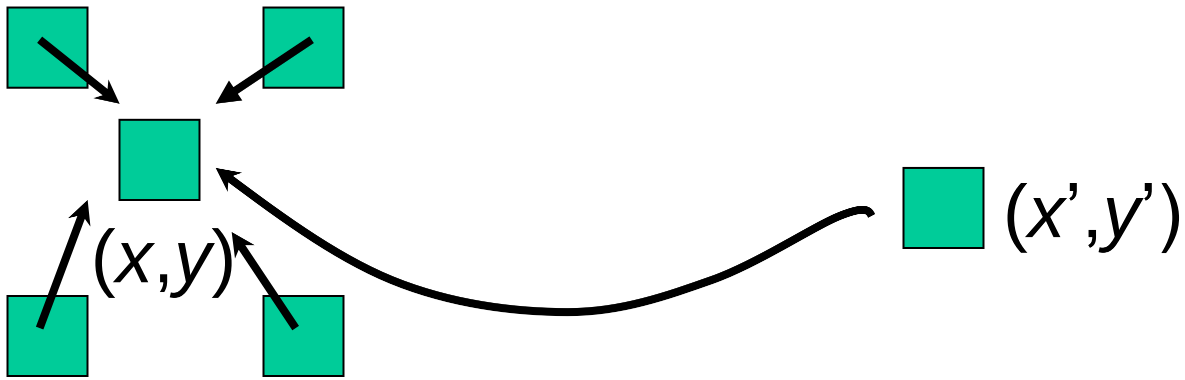
In forward mapping, the coordinate (*x'*,*y'*) is determined depending on the coordinate (*x*,*y*). When (*x'*,*y'*) are not integers, the nearest integer is used.



Forward mapping

Reverse Mapping

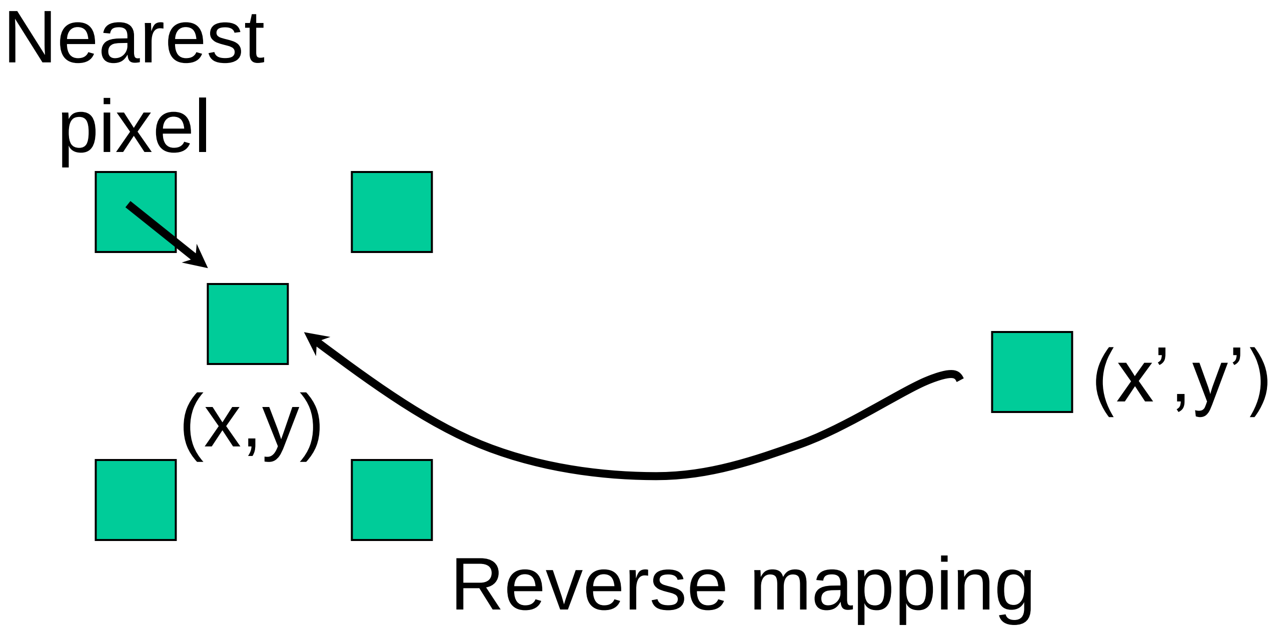
In the case of reverse mapping, we need to compute the values at (x,y) that corresponds to (x',y'). When (x,y) are not integers, we refer to the value(s) of adjacent pixel(s). This process is called *interpolation*.



Reverse mapping

Nearest Neighbor Interpolation (NNI)

The simplest and fastest interpolation method is to use the value of the nearest pixel.



Bilinear Interpolation

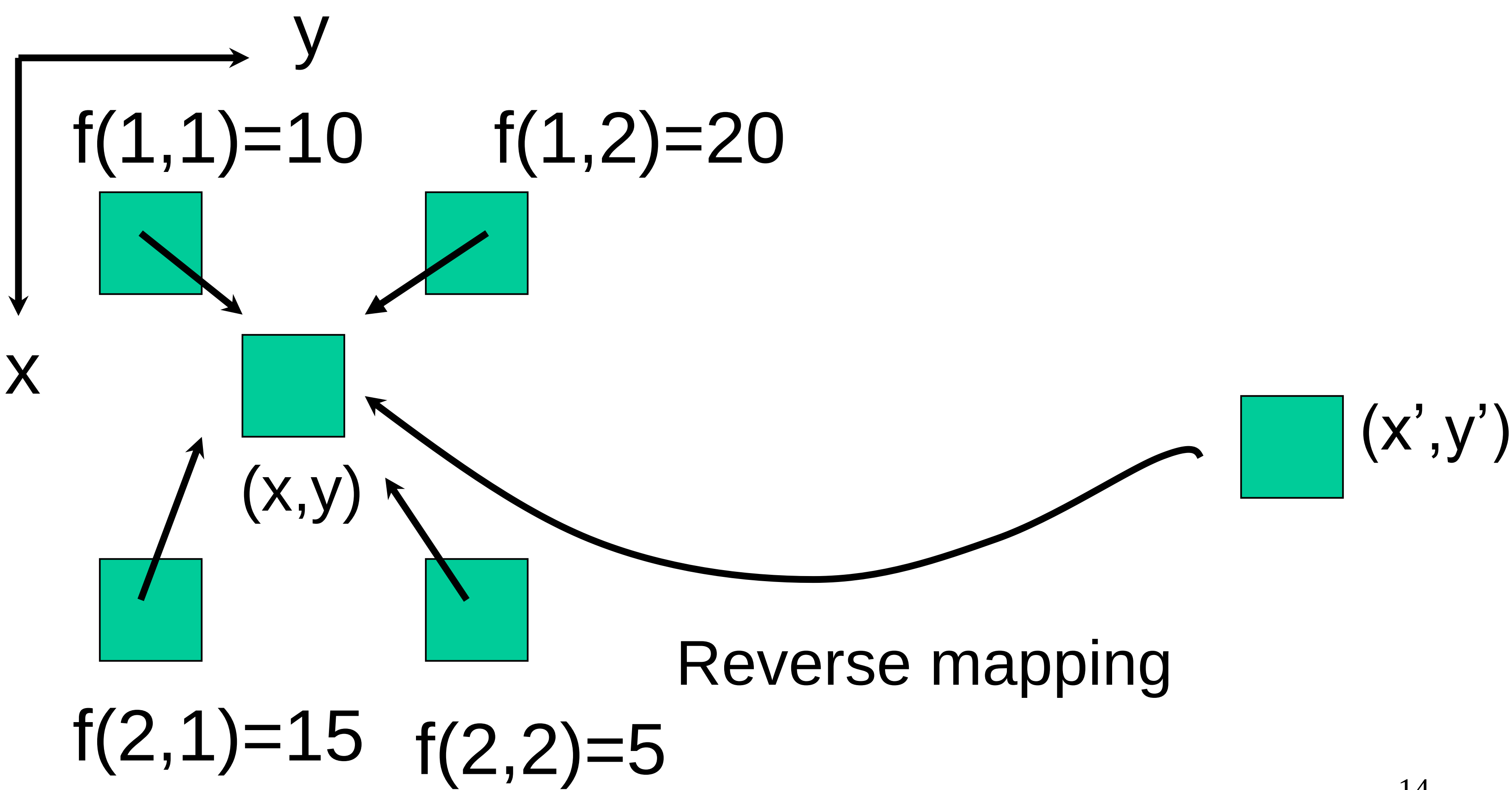
A more smooth interpolation can be achieved by combining the values of four adjacent pixels.

Weights are determined by the inverse of the distances between (x,y) and the four pixels.

Reverse mapping

Exercise 2

Calculate the value at (x,y)=(1.2,1.5) by bilinear interpolation.



Exercise 3

Referring to the figure below, derive the gray level at (x’,y’) by bilinear interpolation.

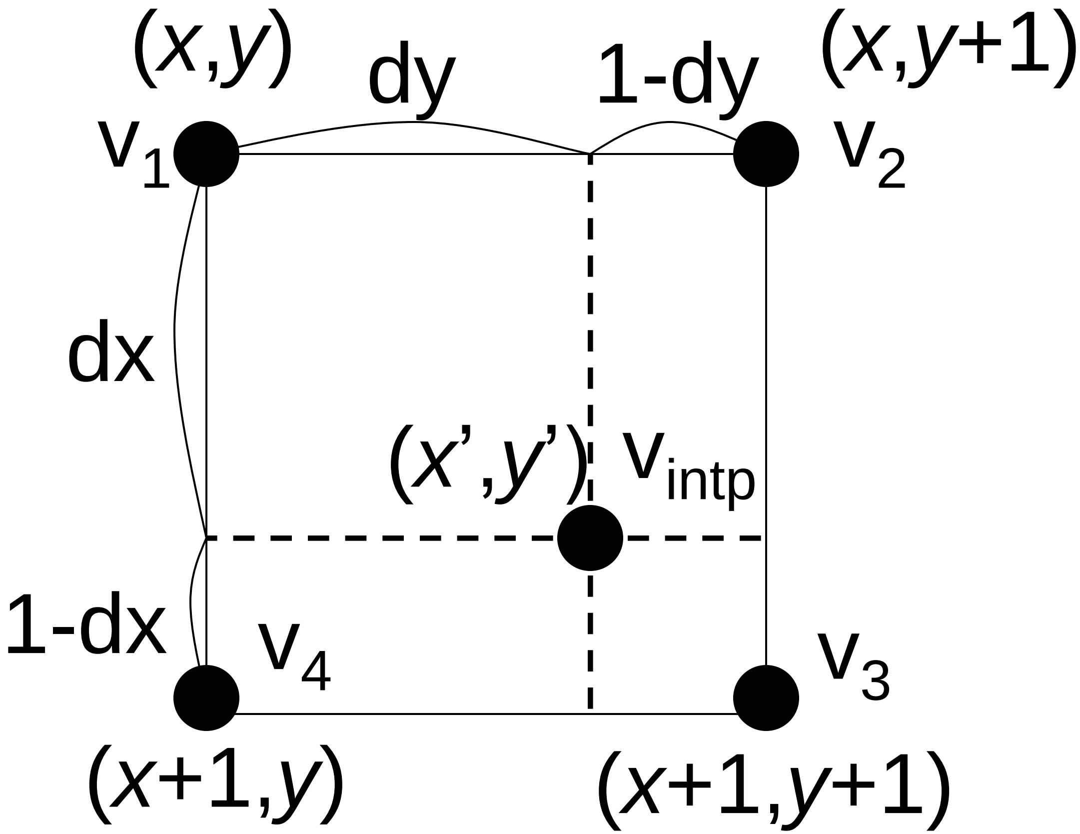


Image Rotation by Forward Mapping

Rotation matrix is given by

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

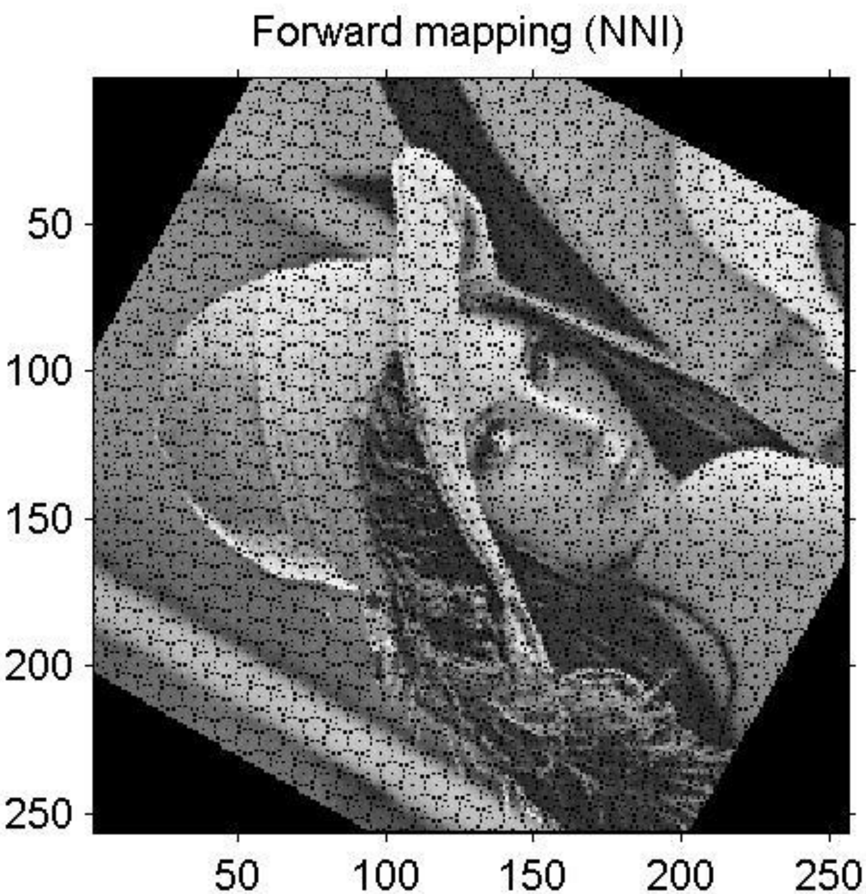
This equation relates the coordinates between input and output images, and can be used for forward mapping directly.

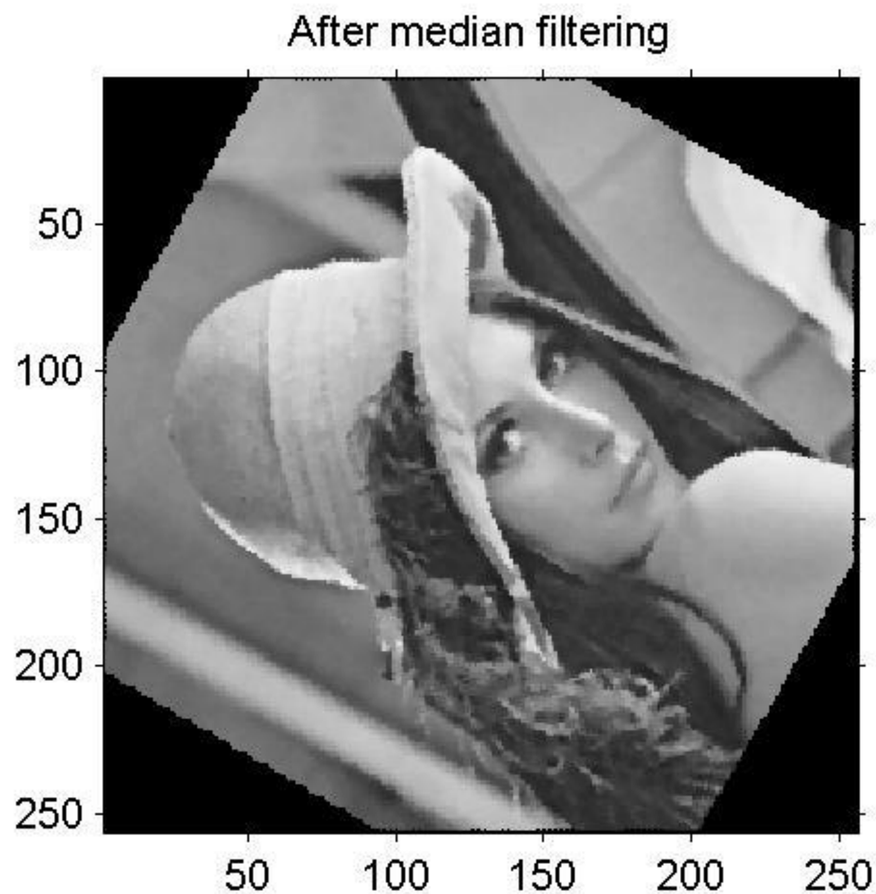
Mapping Function

If you want to rotate an image about its center (x_c,y_c), the mapping function needs to be modified into

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

Rotated Image by Forward Mapping





A Problem of Forward Mapping

Forward mapping often leaves holes in the resultant image. This problem is exacerbated when an image is stretched to a great extent.

If holes are small, they can be filled by the median filter at the sacrifice of some image details.

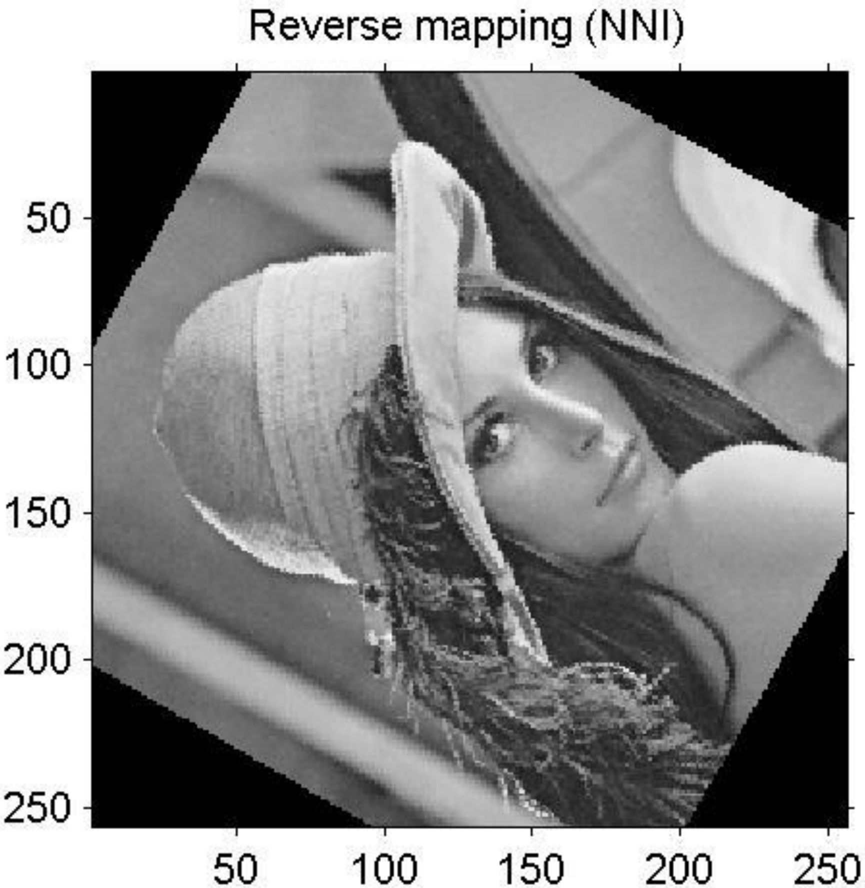
Image Rotation by Reverse Mapping

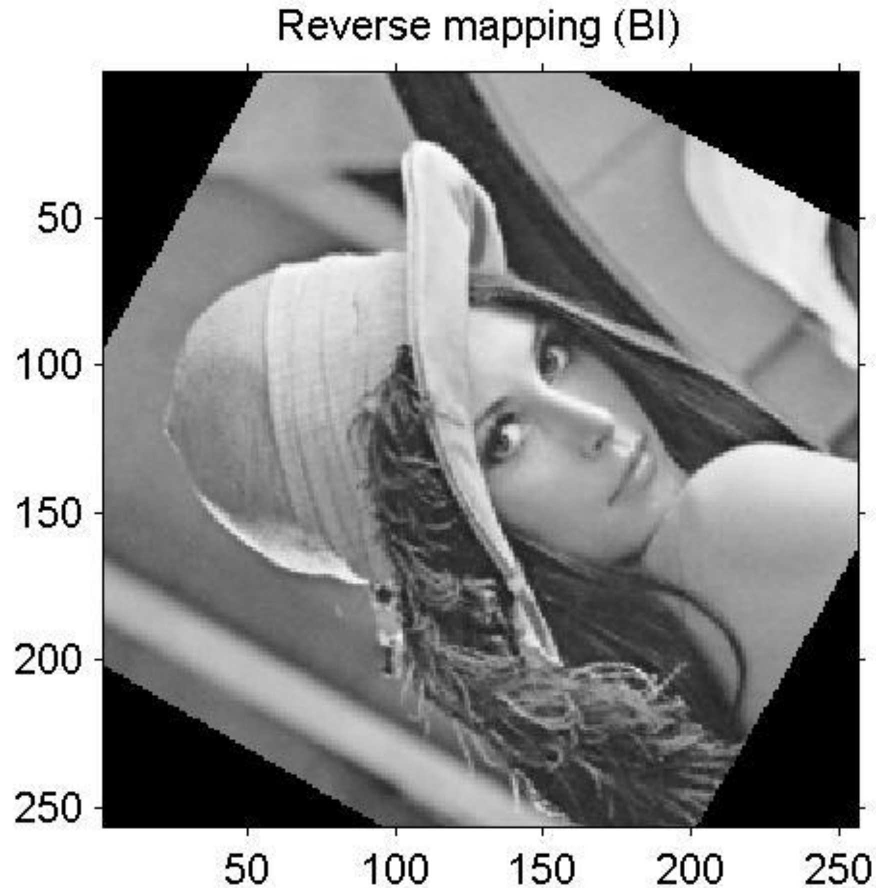
To implement reverse mapping, we need to derive the inverse of a mapping function.

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

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

Rotated Image by Reverse Mapping





No holes any more. Note also the difference between two interpolation methods.

Image Scaling by Forward Mapping

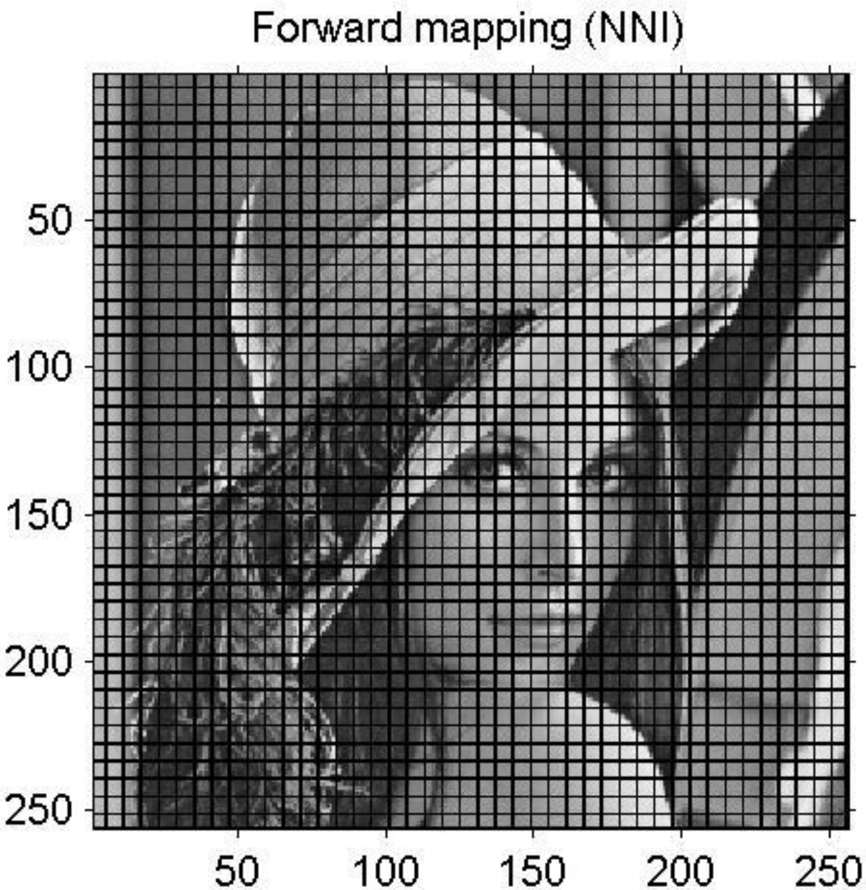
Scaling matrix is given by

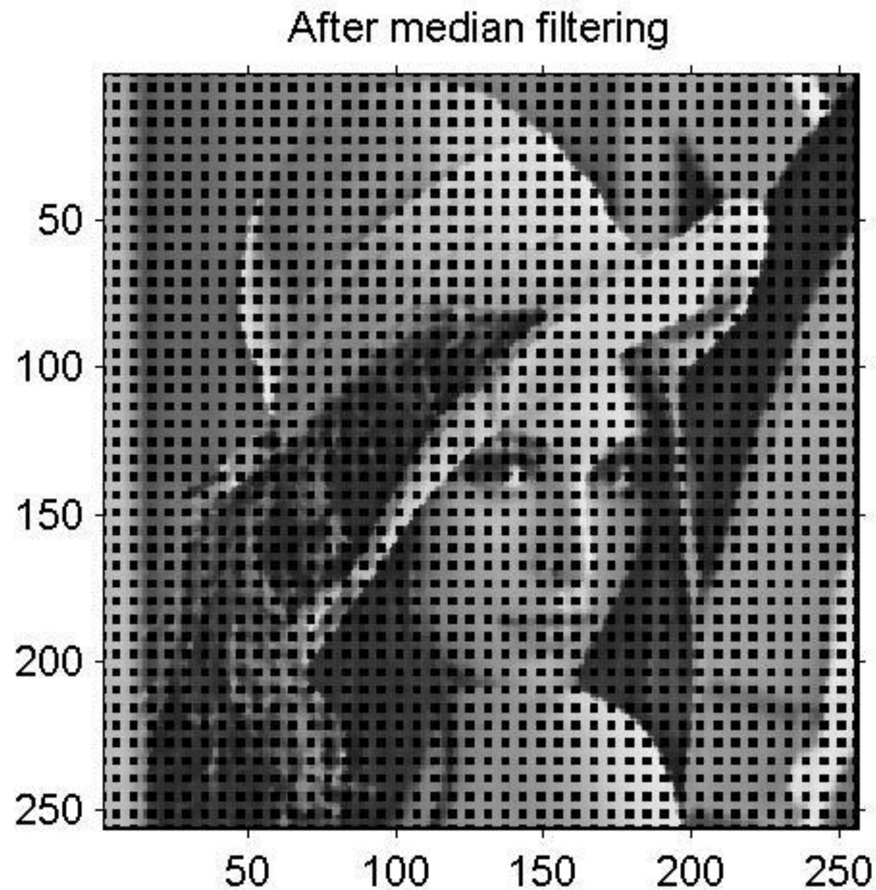
$$\begin{pmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{pmatrix} = \begin{pmatrix} \boldsymbol{m} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{n} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix}$$

where ***m, n*** denote the scaling (zooming) factor. To zoom an image about its center, the mapping function will be

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} m & 0 \\ 0 & n \end{pmatrix} \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}$$

Enlarged Image by Forward Mapping



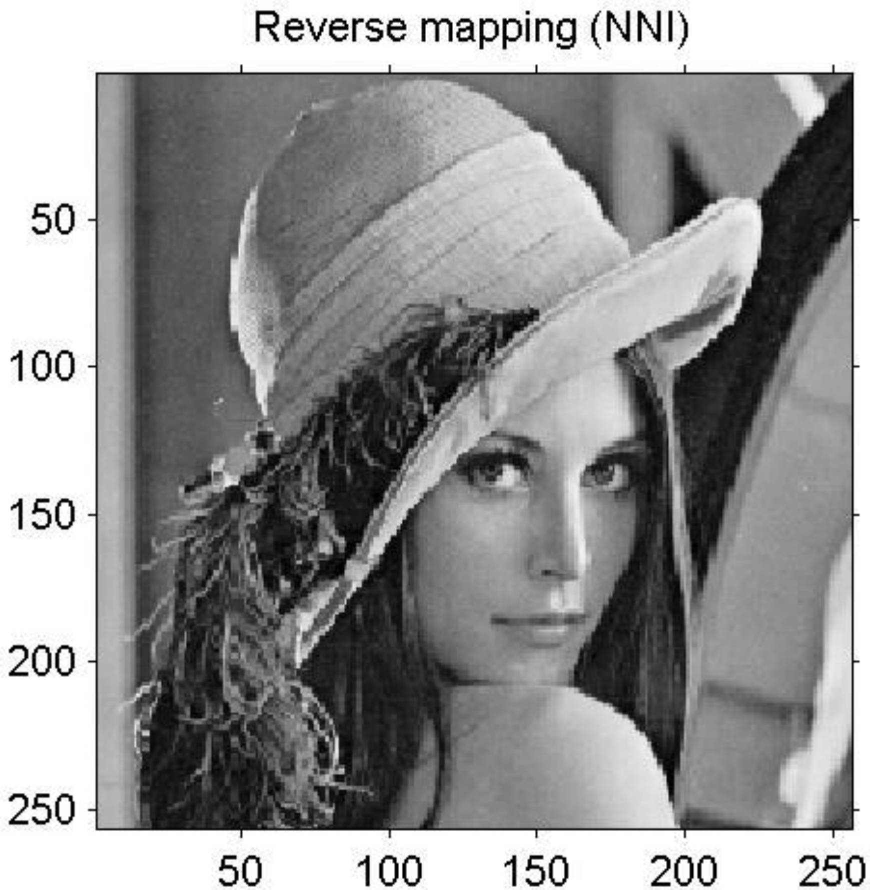


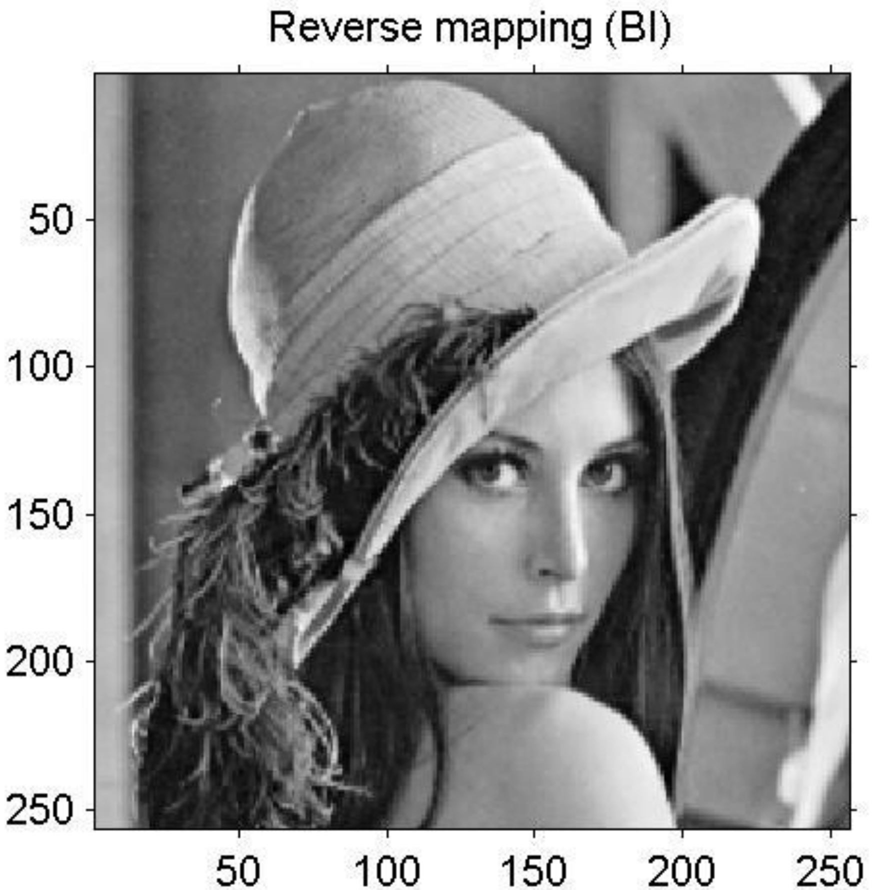
m=1.2 (20% enlarged)

Exercise 4

Show the mapping function for image scaling (zooming) by reverse mapping.

Enlarged Image by Reverse Mapping





m=1.2

Note also the difference between two interpolation methods.

Exercise 5

We want to shift an image f(x,y) by (α, β), magnify it m times, and then rotate it by θ (rad). Express the output coordinates (x',y') in a consolidated matrix form. Express also (x',y') when the order of the three operations is reverse.

Affine Transform

Translation, rotation, and scaling can be expressed with a single equation.

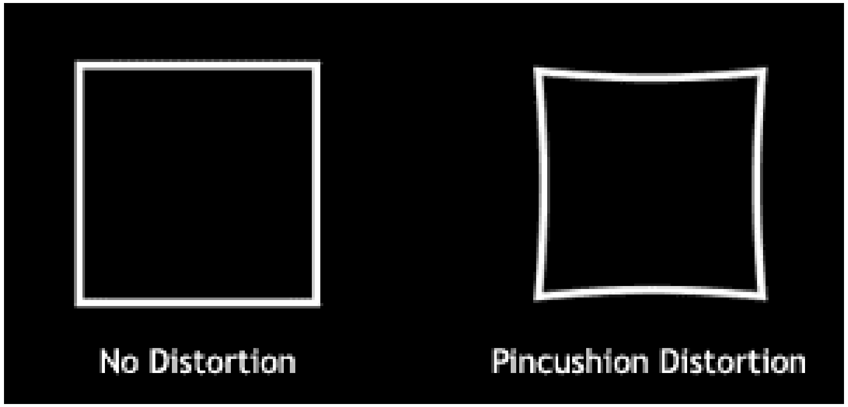
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} A & B \\ D & E \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} C \\ F \end{pmatrix}$$

or

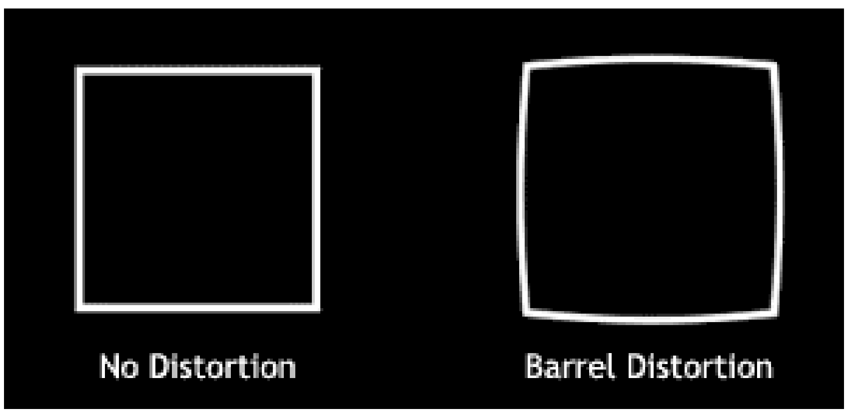
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} A & B & C \\ D & E & F \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

This is called *Affine Transform*.

Radial Distortion



Example of Pincushion Distortion



Example of Barrel Distortion

Pincushion Distortion

Pincushion distortion may be expressed by

$$\begin{cases} x' = x + K \cdot (x - x_0) \cdot \left\{ (x - x_0)^2 + (y - y_0)^2 \right\} \\ y' = y + K \cdot (y - x_0) \cdot \left\{ (x - x_0)^2 + (y - y_0)^2 \right\} \end{cases}$$

where x_0, y_0 are the image center, K is a constant factor to determine the degree of distortion.

Pincushion Distortion

Input image

- Forward mapping
- $K = 0.000005$
- Median filter = 5×5

Pincushion distortion

After median filtering

Exercise 6

Referring to the equations of pincushion distortion, derive the equations for generating barrel distortion.

Barrel Distortion

Input image

- Forward mapping
- $K = 0.000005$
- Median filter = 5×5

Barrel distortion

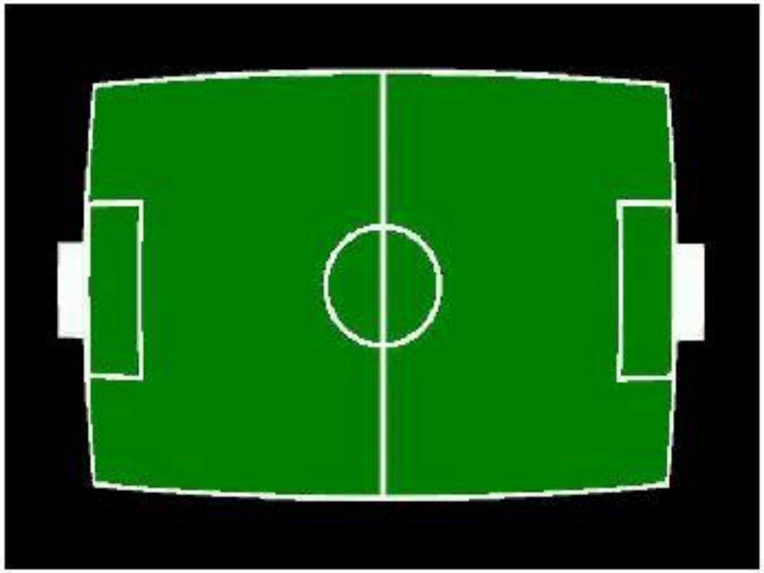
After median filtering

Image Calibration

By adding a reverse distortion, an unwanted image distortion can be corrected.

- Reverse mapping
- $K = 0.000001$

Barrel-distorted soccer field



Calibrated soccer field

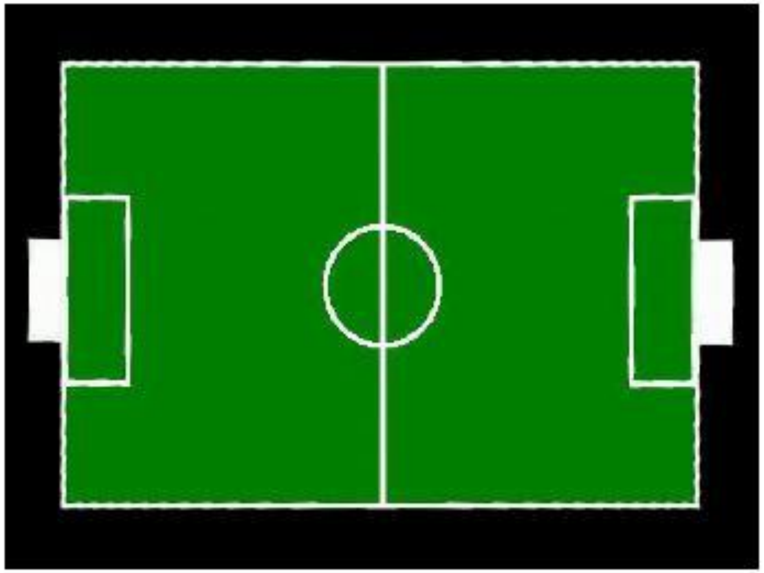


Image Warping

The mapping function for the warp of a quadrilateral region is given by

$$x' = c_1 x + c_2 y + c_3 xy + c_4$$

$$y' = c_5 x + c_6 y + c_7 xy + c_8$$

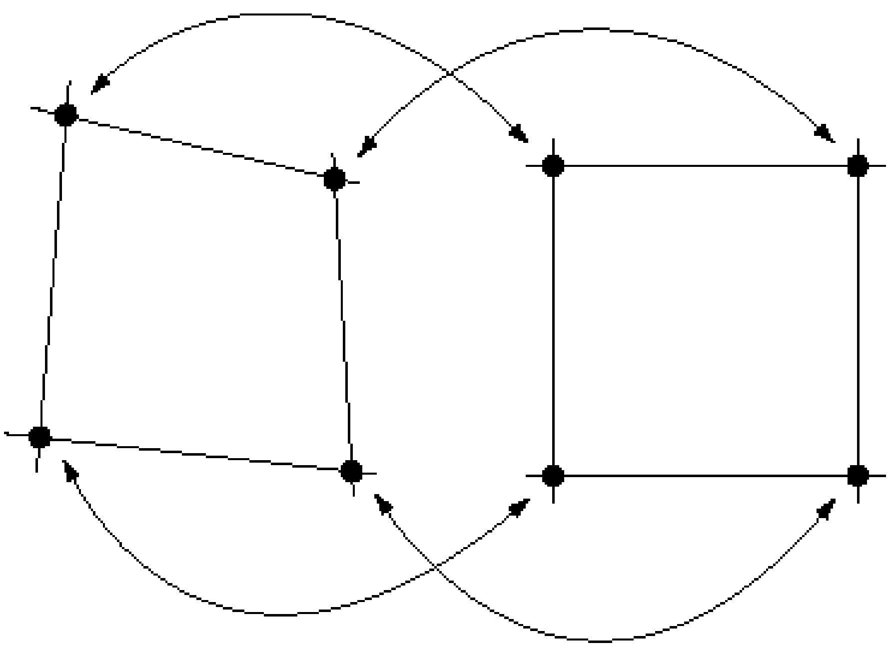
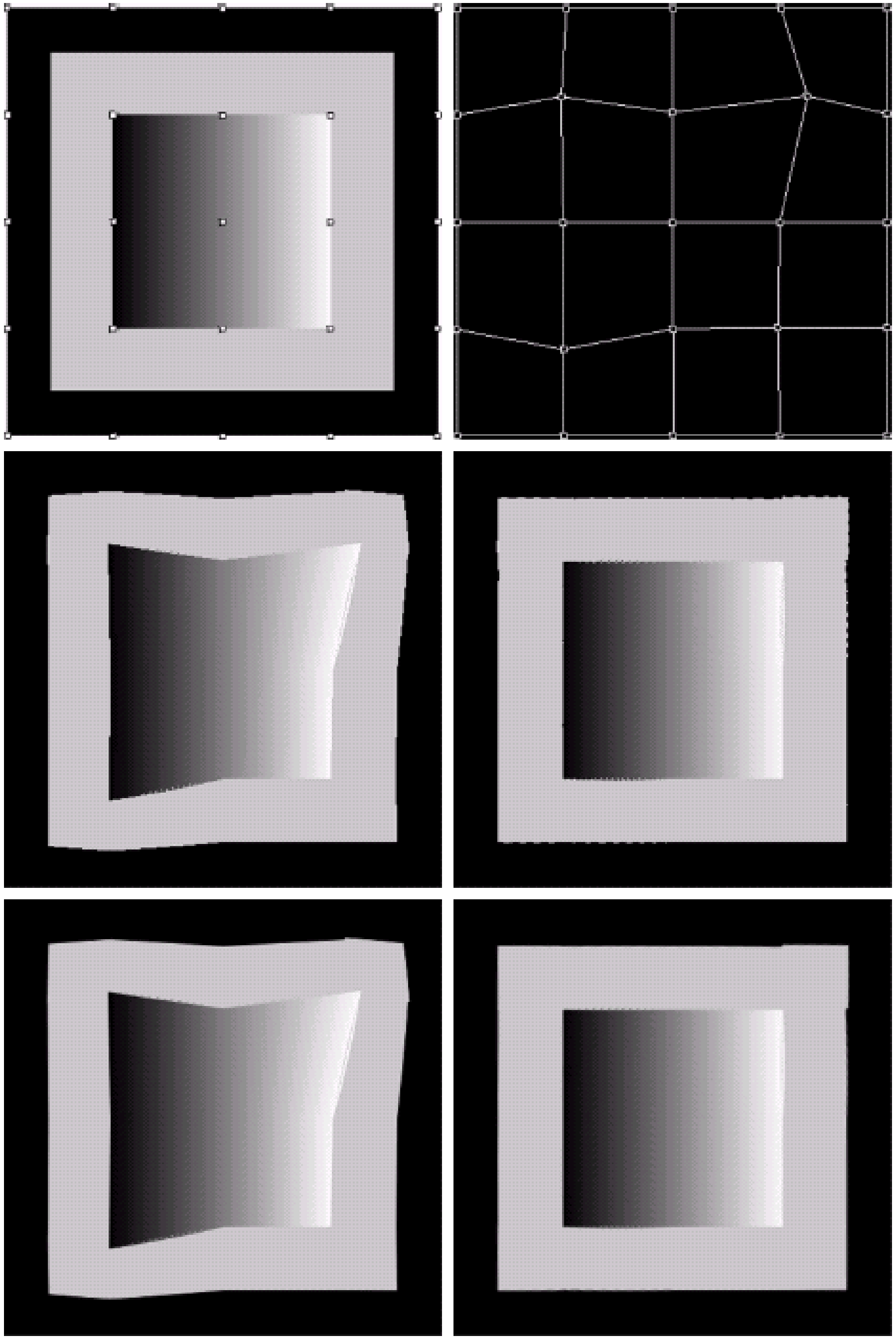


FIGURE 5.32 Corresponding tiepoints in two image segments.

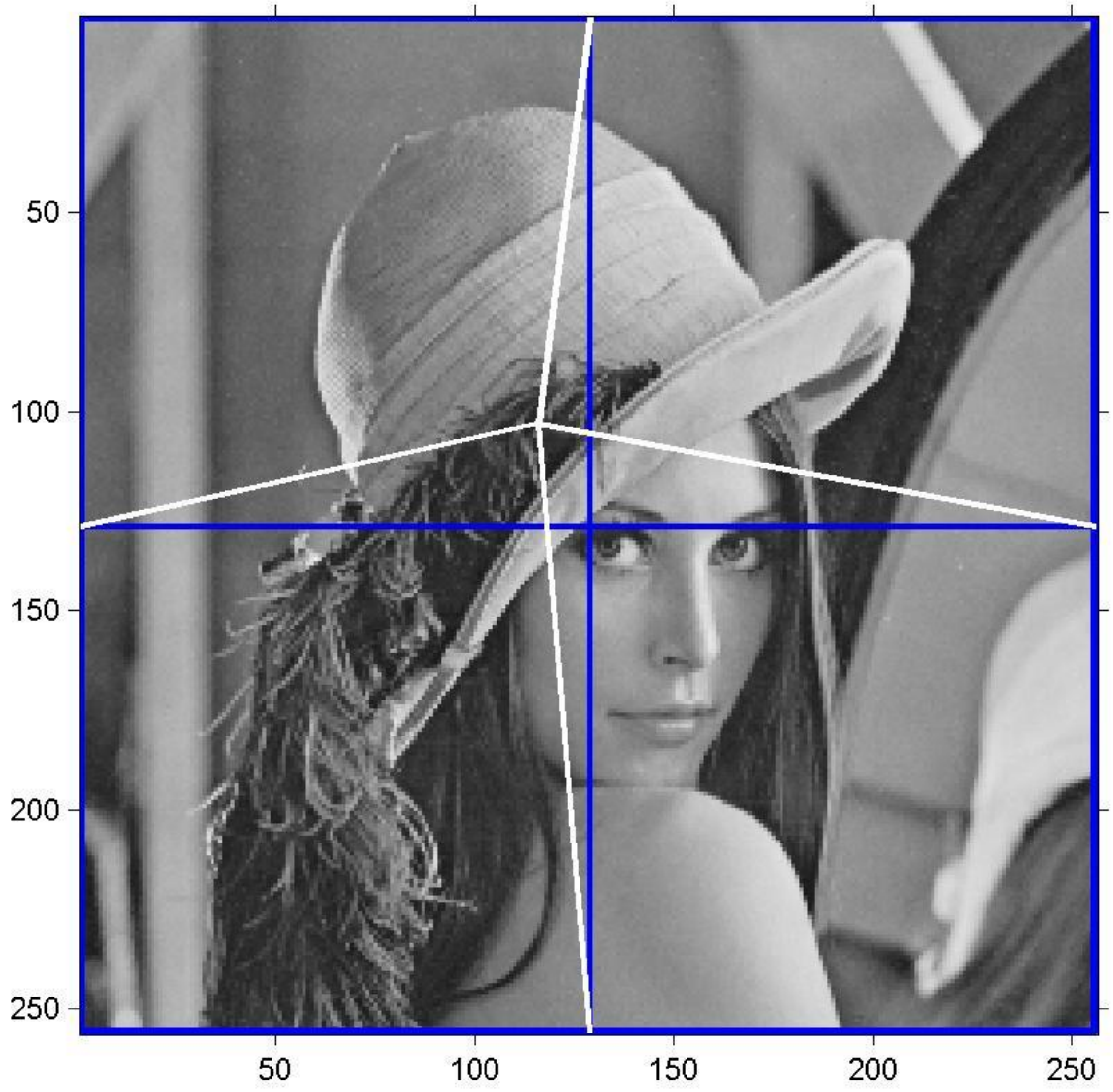


(a) Image with control points.
(b) Control points after distortion.
(c) Distorted image, using NNI.
(d) Restored image.
(e) Distorted image, using BI.
(f) Restored image.

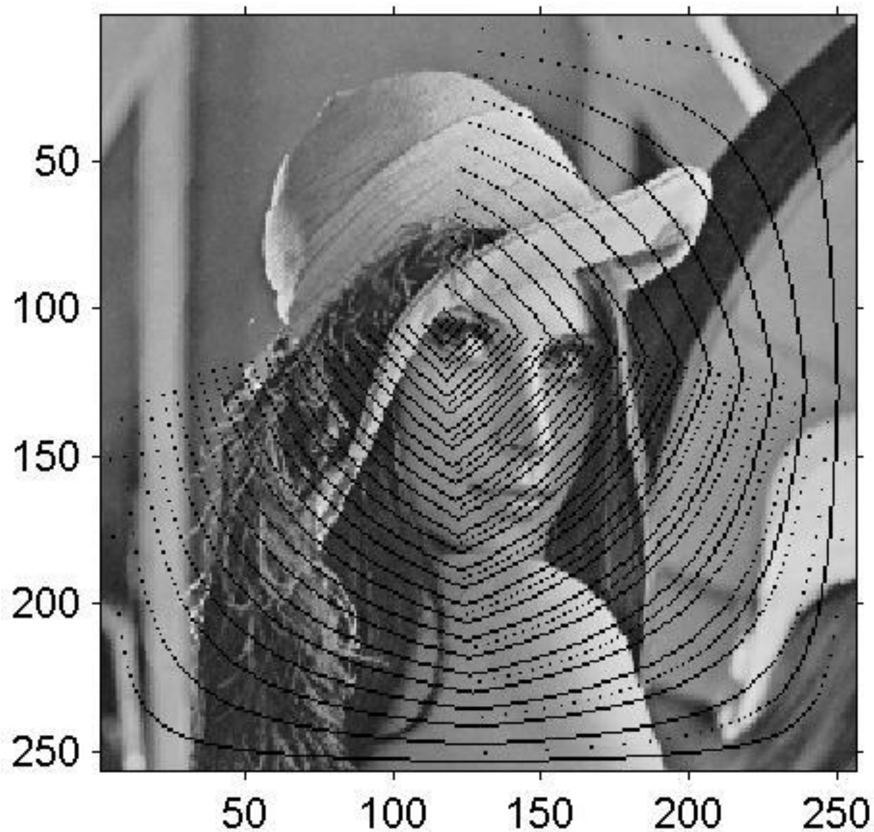
Exercise 7

Assume that the coordinates (x,y) of four control points and their new coordinates (x',y') after distortion are known.
Obtain the coefficients c_1 to c_8 using least-squares method.

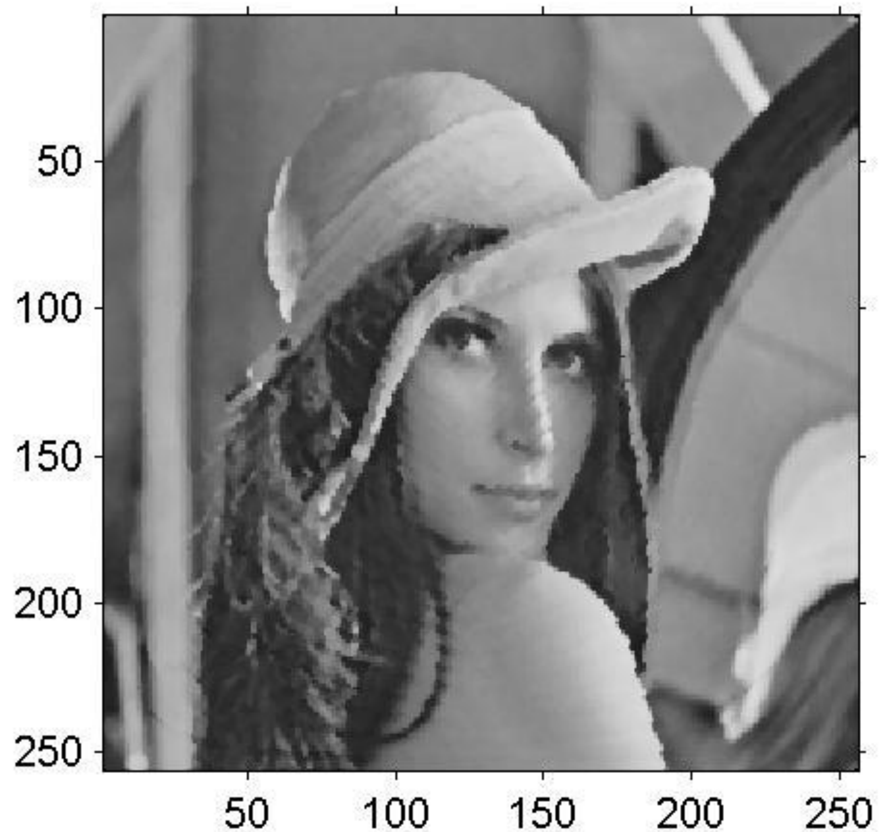
Image Warping Demo



Lena Image Warped by Forward Mapping



Before Median filtering



After Median filtering

Assignment from Chapter 5

1. Generate two symmetrical facial images of yours. <u>Show your own code.</u>

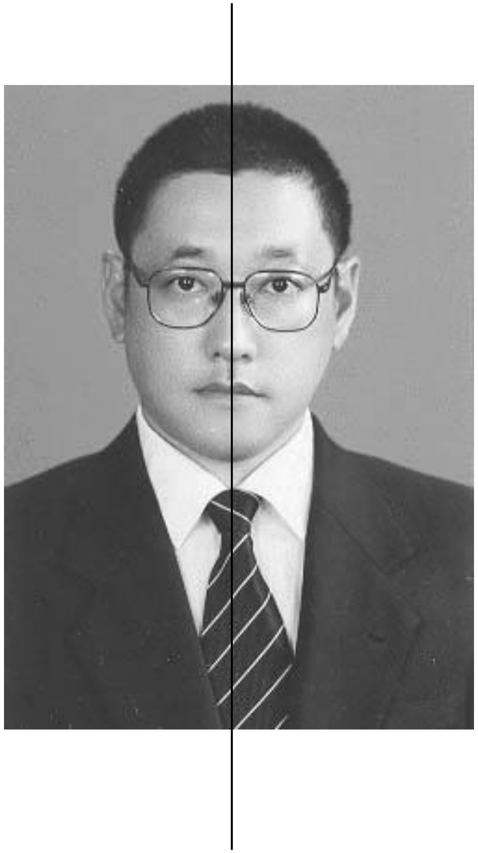
   Setting this option recommended:
   >> iptsetpref('ImshowAxesVisible','on'))

2. Rotate (or resize) one of the symmetrical facial image by an arbitrary angle (or magnification rate, M) using NNI and BI methods. Discuss the differences between two resultant images.

   ```
   >> buf1=imrotate(img,30,'nearest or bilinear');
   >> buf2=imresize(img,M,'nearest or bilinear');
   ```

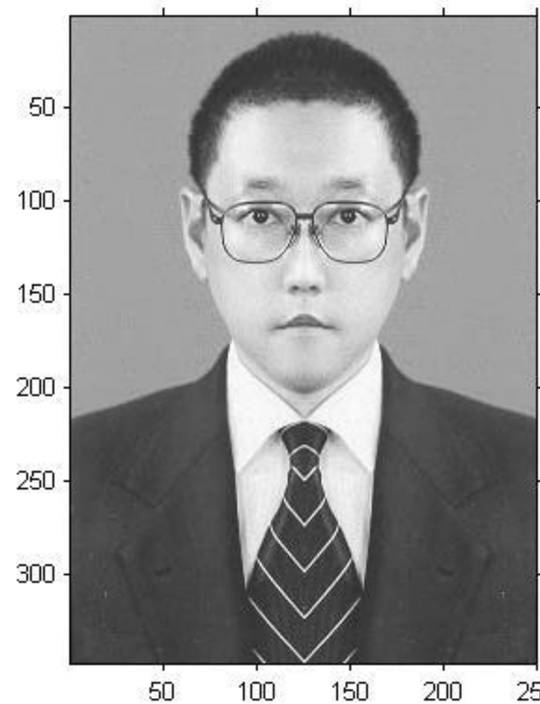
Symmetrical Facial Images

Symmetric axis



Input image

Symmetrical image 1



Symmetrical image 2