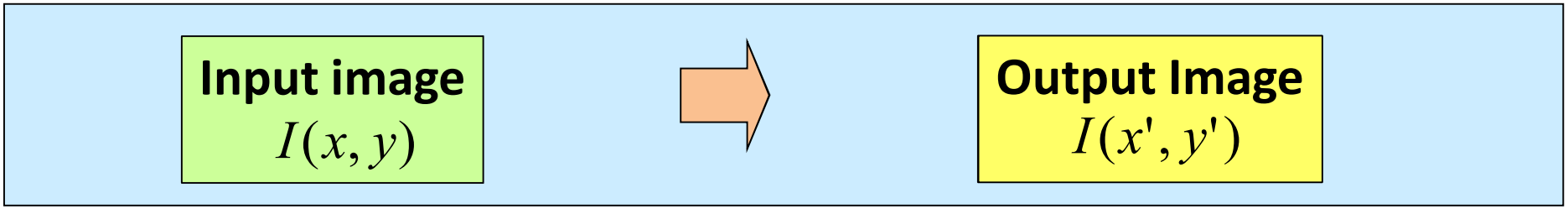
Mahidol University *Wisdom of the Land*

Chapter 5

Geometric Transformation

Geometric Transformation



Geometric transformation : this is the basic process for image processing involves transforming the positions of pixels in an image.

Geometric transformation operations **:**

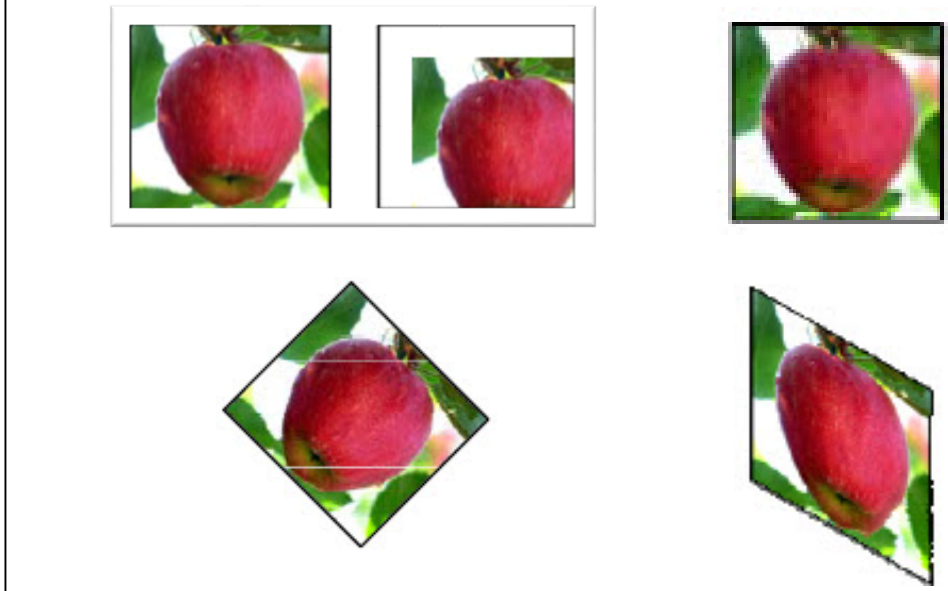
a) Translation
b) Scaling
c) Rotation
d) Shearing

Geometric Transformation

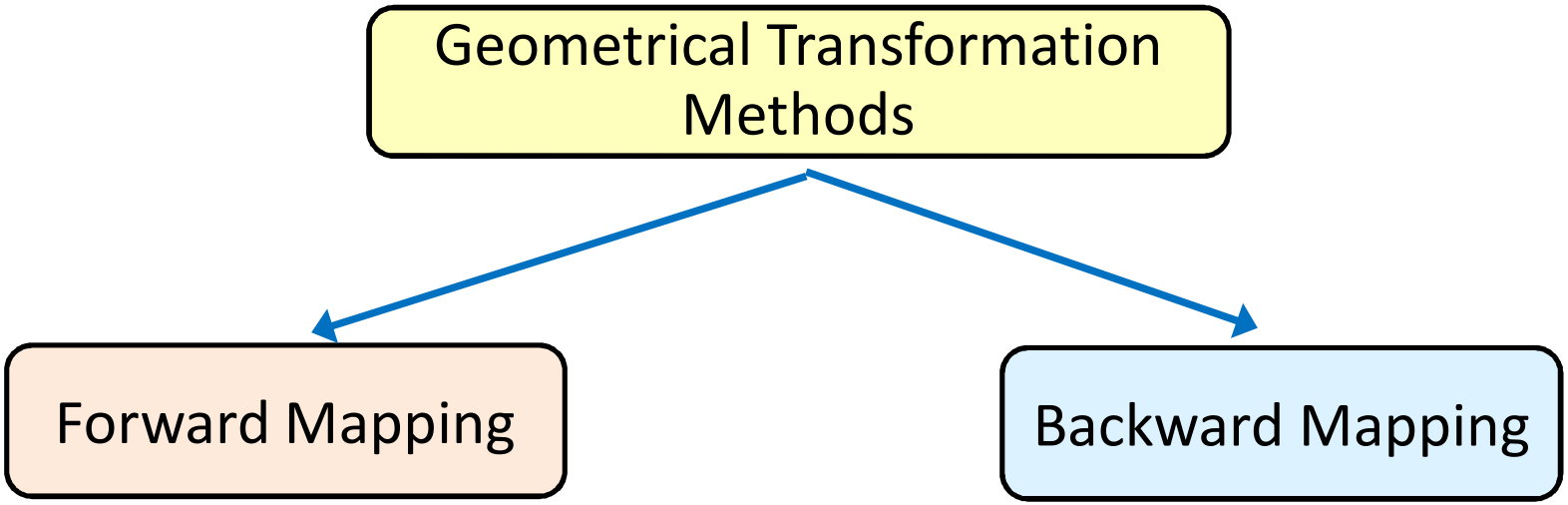
Input image $I(x, y)$ → **Output Image** $I(x', y')$

Geometric transformation operations :

a) Translation
b) Scaling
c) Rotation
d) Shearing



Geometric Transformation



Geometric Transformation Parameters

- Forward Mapping

- The forward mapping is the process of iterate mappings over each pixel in an input image, transforms new coordinates for each pixel, and assigns its value to the new position in an output image.

Forward mapping has two major problems :

- The first problem of forward mapping process is two or more different pixels within the input image can be transformed into the same pixel in the output image, bring into question of how to combine multiple input pixel values into a single output pixel value.
- The second problem is some output pixels may not be assigned. This leads to holes where no value is assigned to a pixel in the output image.

Geometric Transformation Parameters

- Forward Mapping

Translation	$x' = x + T_x$ $y' = y + T_y$
Scaling	$x' = x.S_x$ $y' = y.S_y$
Rotation	$x' = x.\cos\theta - y.\sin\theta$ $y' = x.\sin\theta + y.\cos\theta$
Shearing	$x' = x + y.Sh_x$ $y' = y + x.Sh_y$

Geometric Transformation Parameters

- Backward Mapping

- The backward mapping is the process of iterate mappings over each pixel in the input image, and applies the inverse transformation to assign the position within the input image from which a value must be sampled.
- Although, the assigned coordinates may not be integers and may not lie in the bounds of the output image. However, the output image has no holes.

Geometric Transformation Parameters

- Backward Mapping

Translation	$x' = x - T_x$ $y' = y - T_y$
Scaling	$x' = x.(1/S_x)$ $y' = y.(1/S_y)$
Rotation	$x' = x.\cos\theta + y.\sin\theta$ $y' = -x.\sin\theta + y.\cos\theta$
Shearing	$x' = x + y.(1/Sh_x)$ $y' = y + x.(1/Sh_y)$

Translation

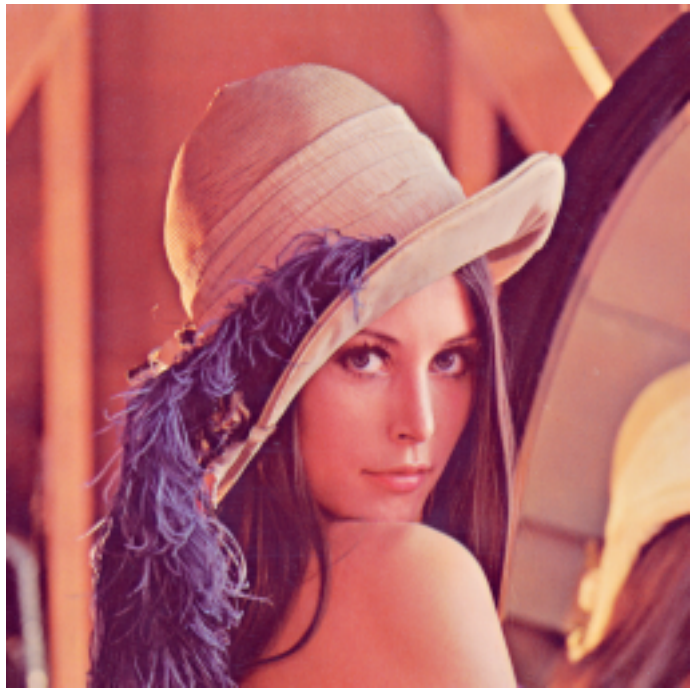
- In order to translate image $I(x, y)$ to image $I(x', y')$ involves translating the position of original pixel to a new place by adding some offset T_x to all the x-coordinates, and adding some offset T_y to all the y-coordinates.

Translation ⇨

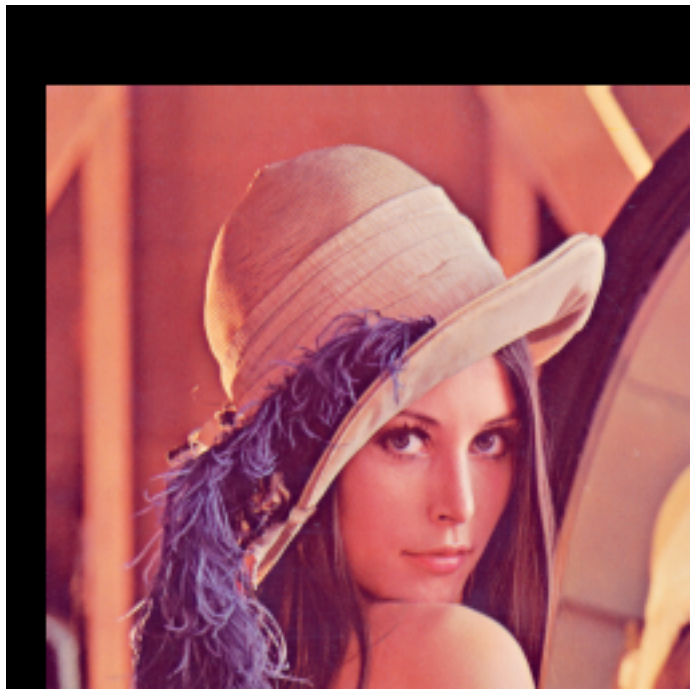
$$x' = x - T_x$$
$$y' = y - T_y$$

- This will shift the original image right T_x and down T_y.
- Use to keep an object of interest in the middle.

Example: Translation



Original image



Translated image with T_x= 30, T_y= 60 using zero padding

Scaling

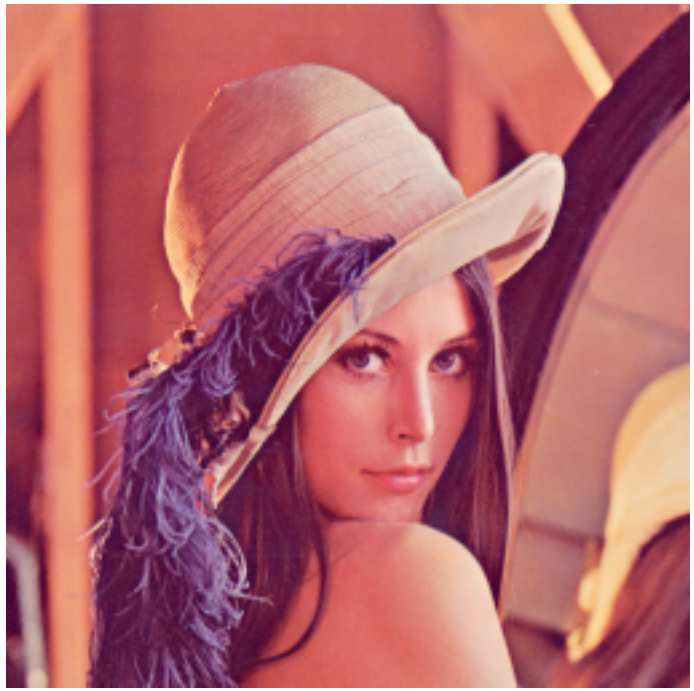
- In order to scale image $I(x, y)$ to image $I(x', y')$ involves shrinking or stretching the position of original pixel by the x-coordinate is shrink or stretch S_x times and the y-coordinate is shrink or stretch S_y times.

Scaling ⇨

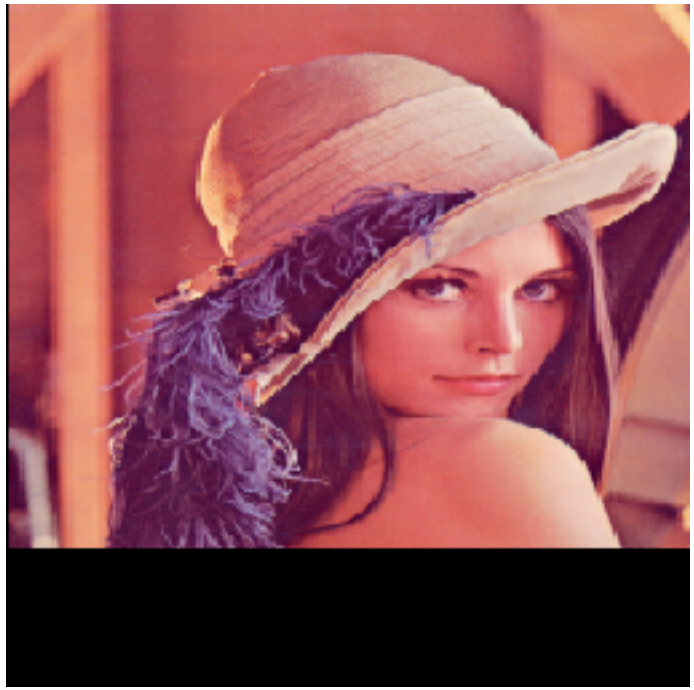
$$x' = x.(1/S_x)$$
$$y' = y.(1/S_y)$$

- This will shrink or stretch the original image by S_x and S_y.
- Use to zoom-out or zoom-in on the object of interest without changing size of output image.

Example: Scaling



Original image



Scaled image with S_x= 1.2, S_y= 0.8 using zero padding

Rotation

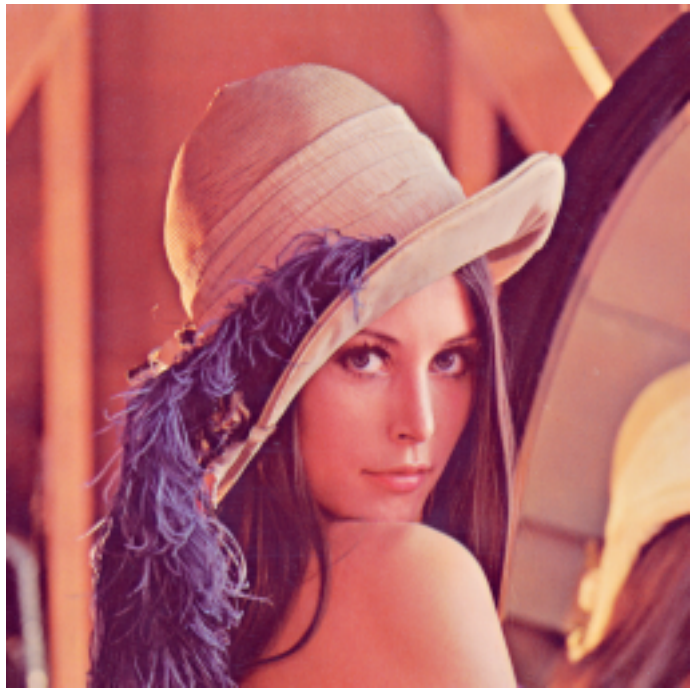
- In order to rotate image $I(x, y)$ around the origin $(0,0)$ to image $I(x', y')$ involves rotating the position of original pixel to a new place by some angle θ.

Rotation

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

- This will rotate image clockwise by θ angles.
- Used to correct orient the object of interest within image.

Example: Rotation



Original image

Rotated image with θ = 5
using zero padding

Shearing

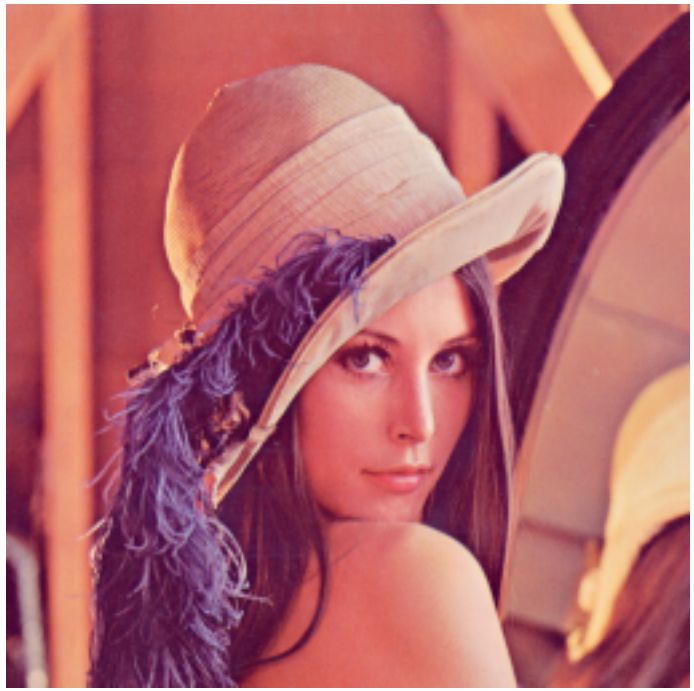
- In order to shear image $I(x, y)$ to image $I(x', y')$ involves tilting image in the x-coordinate with ratio Sh_x and tilting image in the y-coordinate with ratio Sh_y.

Shearing ⇨

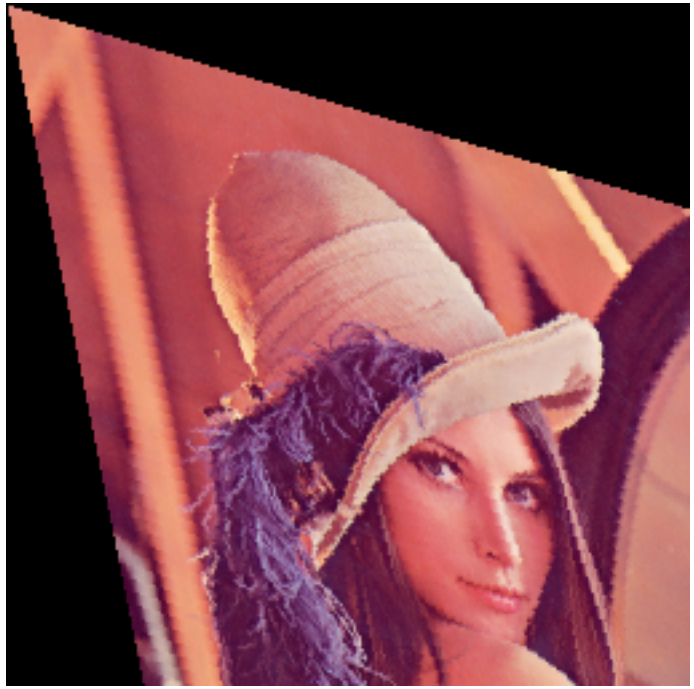
$$x' = x + y.(1 / Sh_x)$$
$$y' = y + x.(1 / Sh_y)$$

- This will shear image in X and Y directions.
- Used to correct image distortions.

Example: Shearing



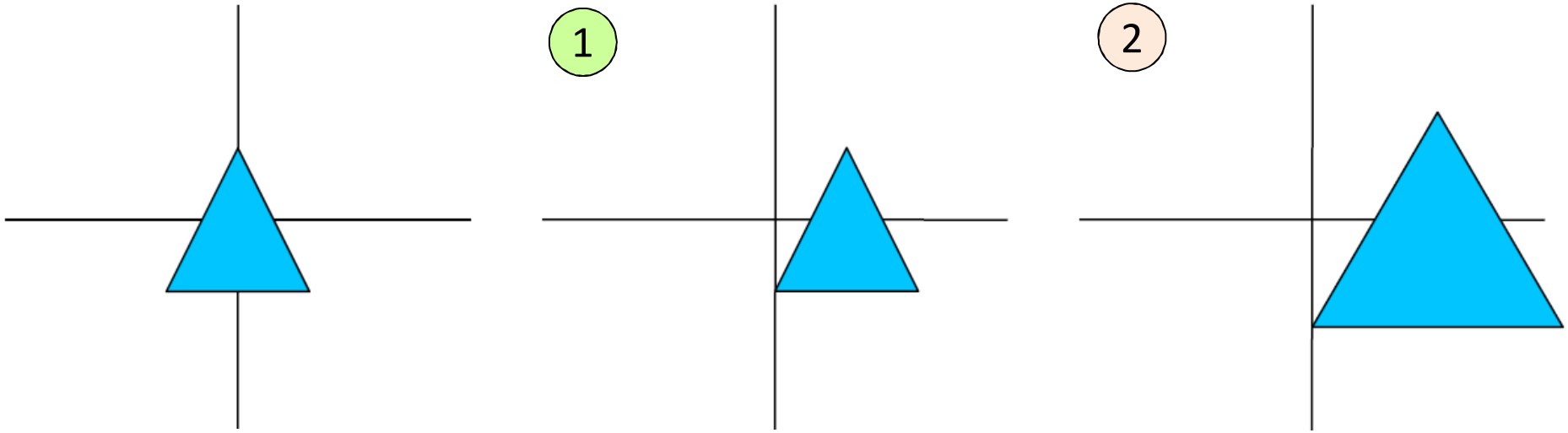
Original image



Sheared image with Sh_x = 0.2, Sh_y = 0.3 using zero padding

Combined Geometric Transformation Operations

Original image → Translation → Scaling → Output image



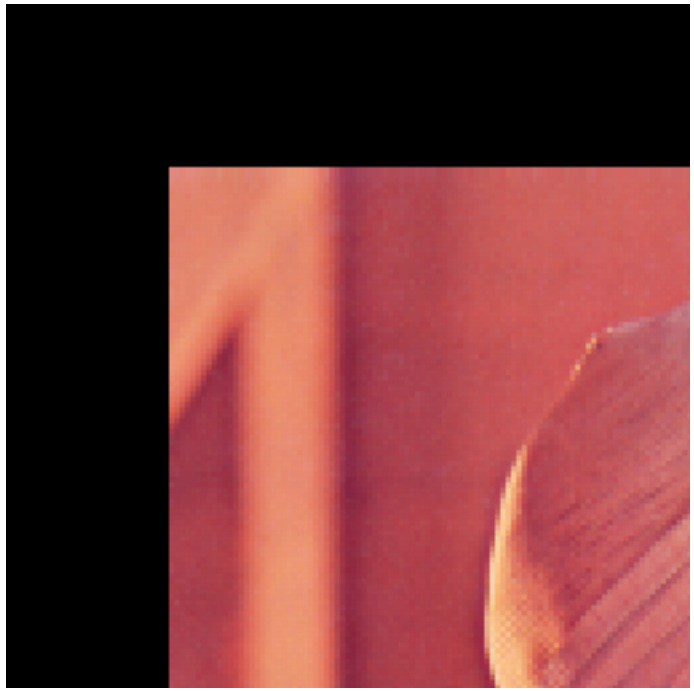
Translation then Scaling

$$x' = (x - T_x) / S_x$$
$$y' = (y - T_y) / S_y$$

Example: Combined Geometric Transformation Operations

Method1 : Translation then Scaling

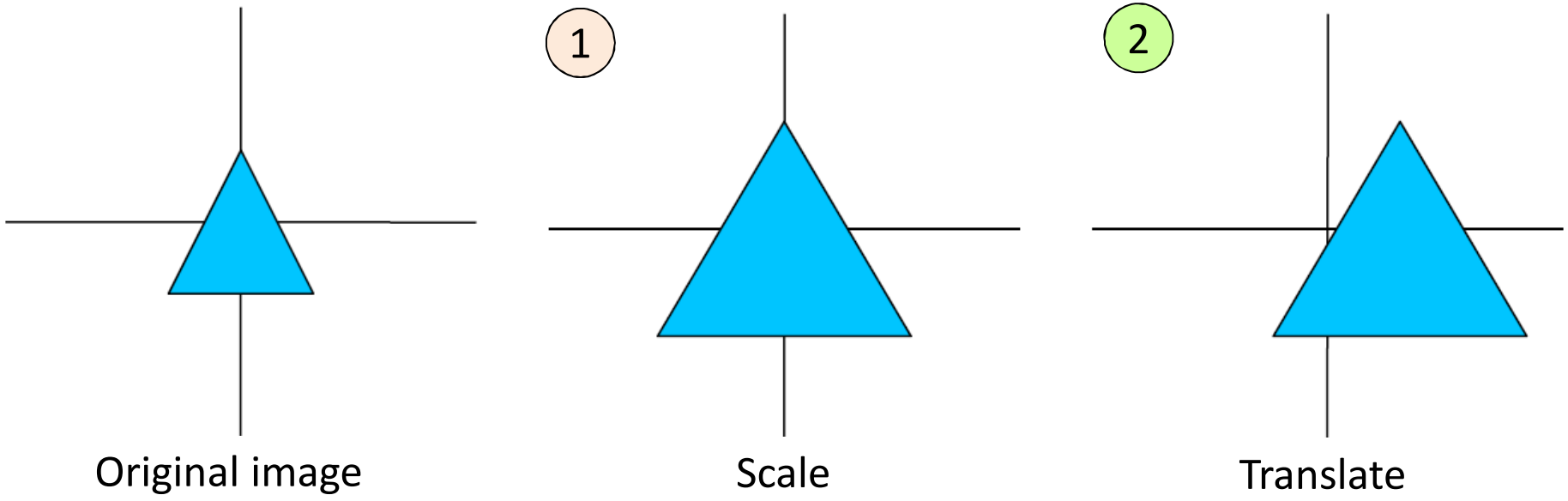
Original image



The resulted image with $T_x = 20$, $T_y = 20$, $S_x = 2$, and $S_y = 2$

Combined Geometric Transformation Operations

Original image → Scaling → Translation → Output image



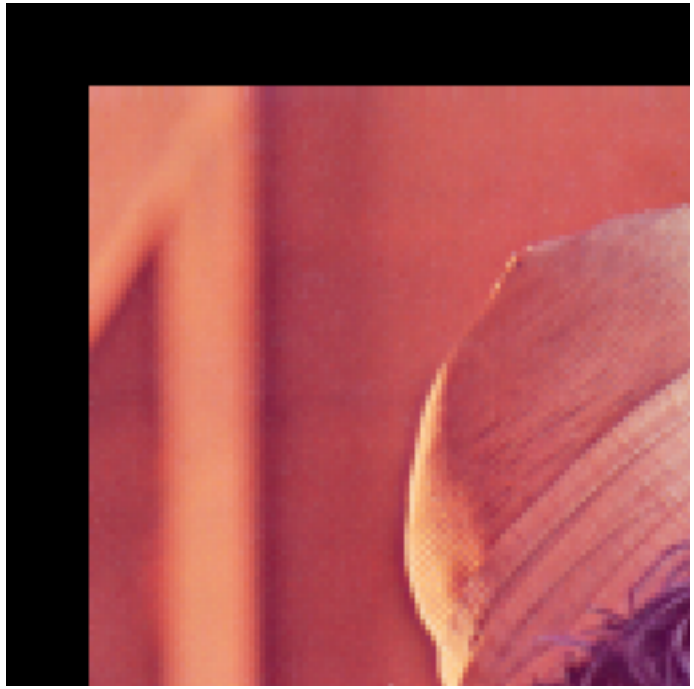
Scaling then Translation

$$x' = (x / S_x) - T_x$$
$$y' = (y / S_y) - T_y$$

Example: Combined Geometric Transformation Operations

Method2 : Scaling then Translation

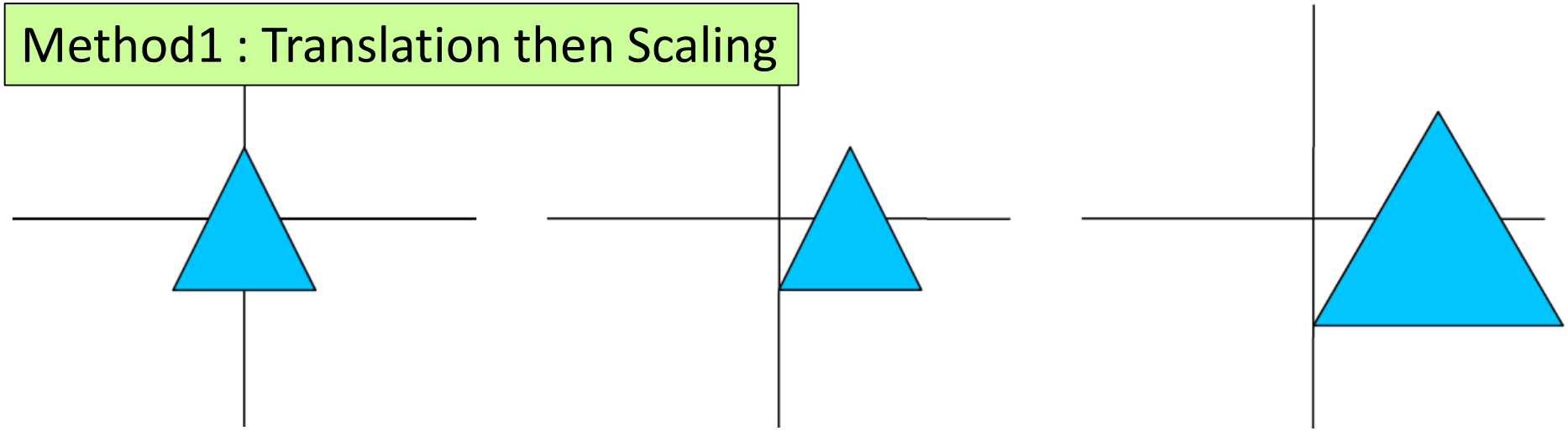
Original image



The resulted image with $S_x = 2$, $S_y = 2$, $T_x = 20$, and $T_y = 20$

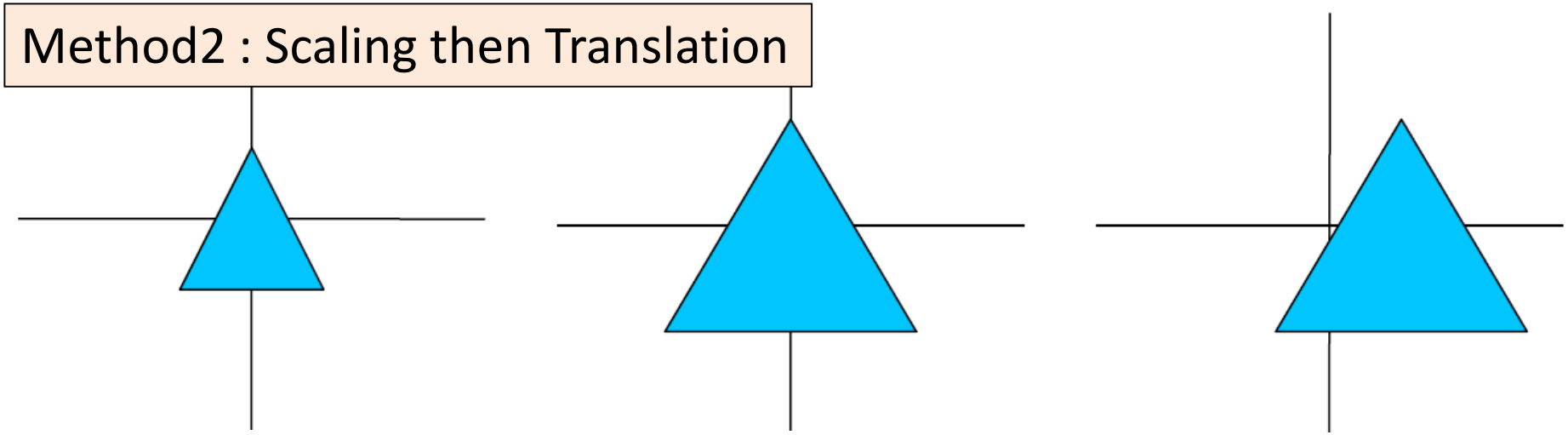
Combined Geometric Transformation Operations

Method1 : Translation then Scaling



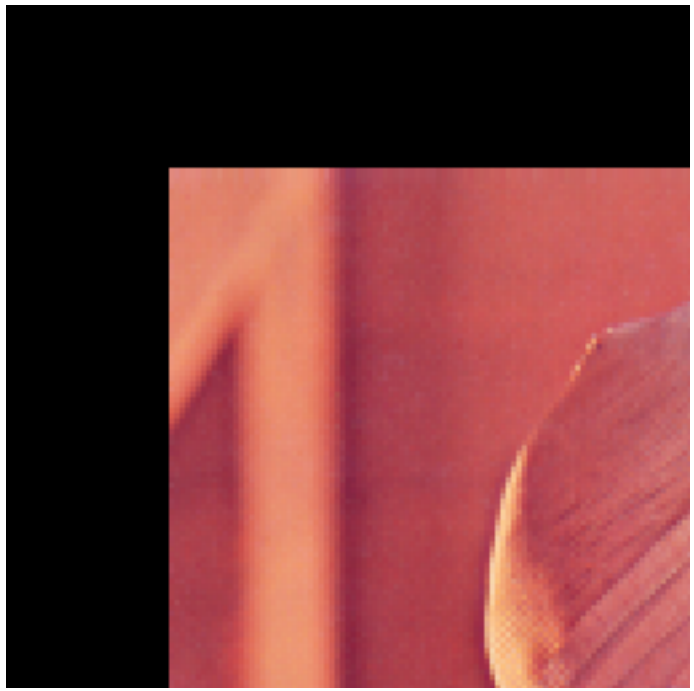
Original image | Translate | Scale

Method2 : Scaling then Translation

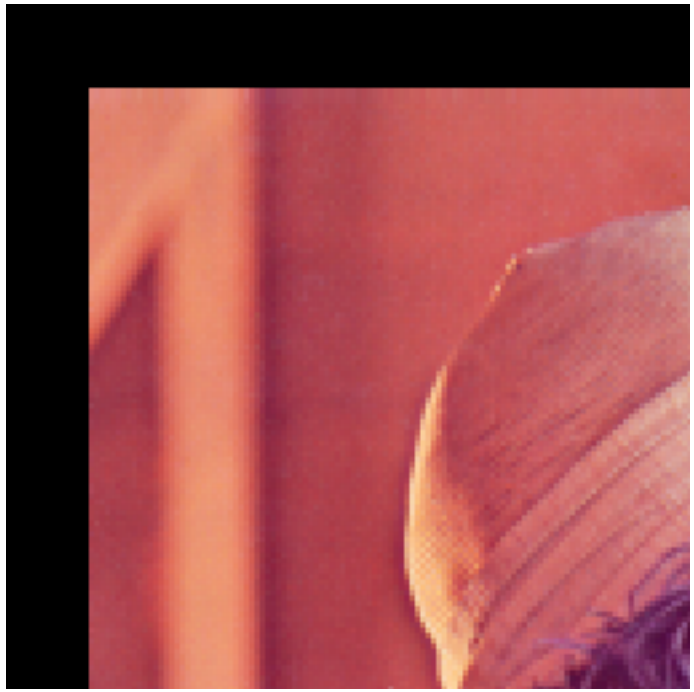


Original image | Scale | Translate

Example: Combined Geometric Transformation Operations



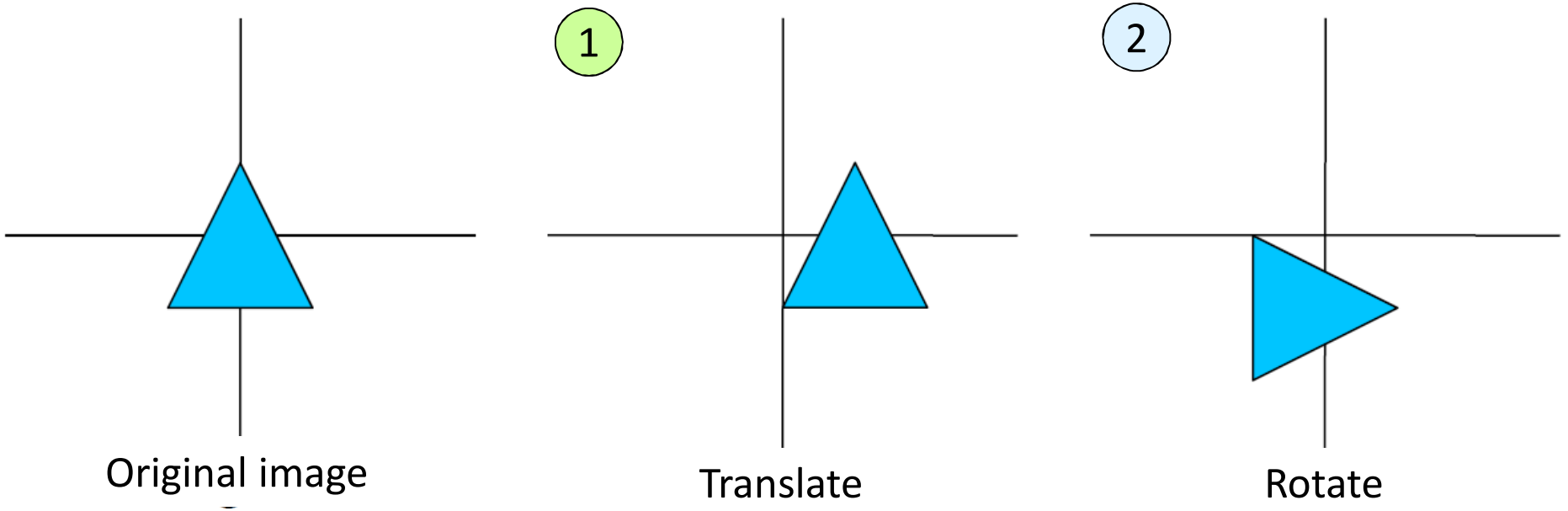
Method1
Translation then Scaling



Method2
Scaling then Translation

Combined Geometric Transformation Operations

Original image → Translation → Rotation → Output image

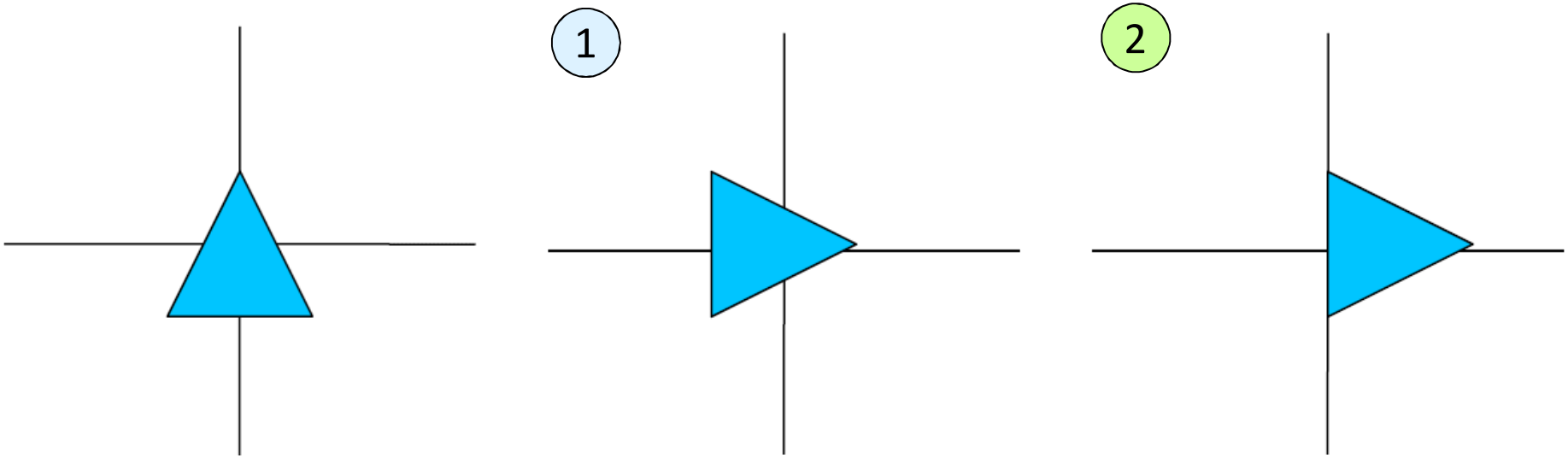


Translation then Rotation

$$x' = (x - T_x).\cos\theta - (y - T_y).\sin\theta$$
$$y' = (x - T_x).\sin\theta + (y - T_y).\cos\theta$$

Combined Geometric Transformation Operations

Original image → Rotation → Translation → Output image

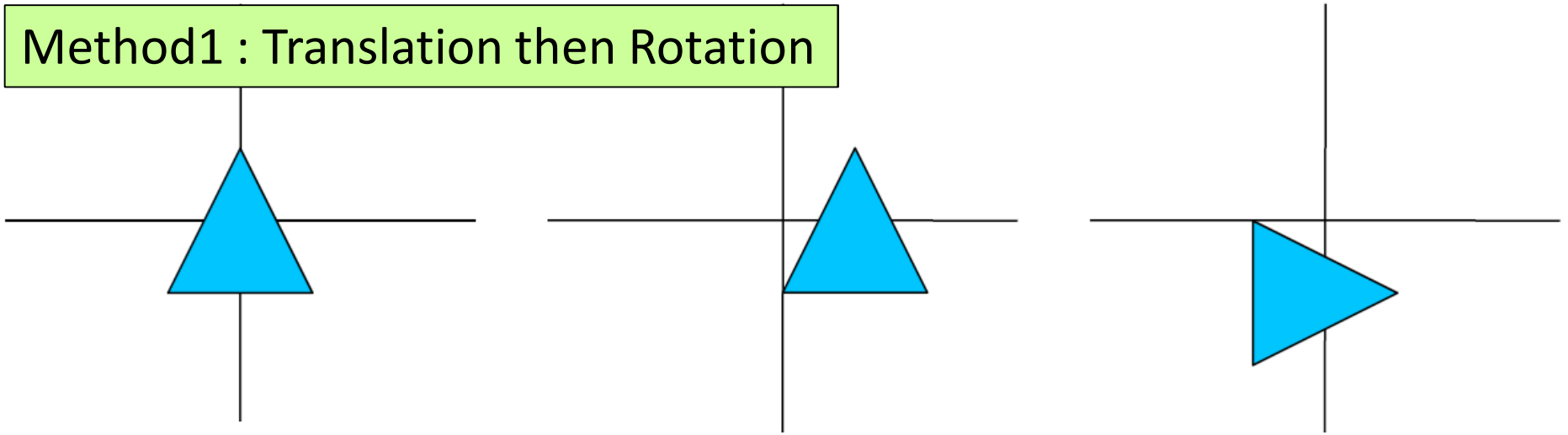


Rotation then Translation

$$x' = (x.\cos\theta - y.\sin\theta) - T_x$$
$$y' = (x.\sin\theta + y.\cos\theta) - T_y$$

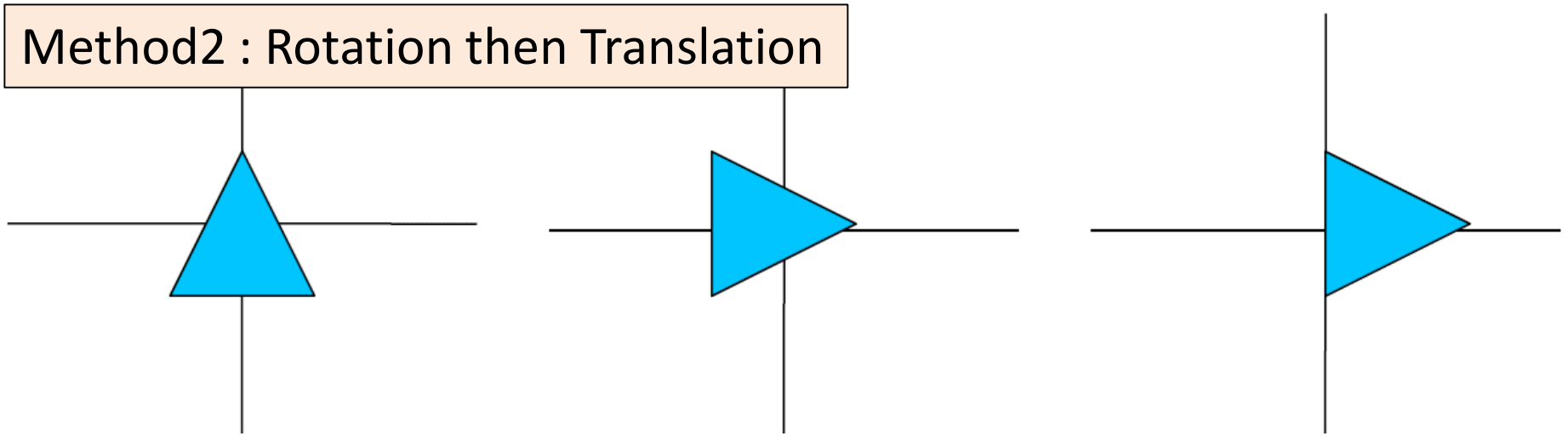
Combined Geometric Transformation Operations

Method1 : Translation then Rotation



Original image | Translate | Rotate

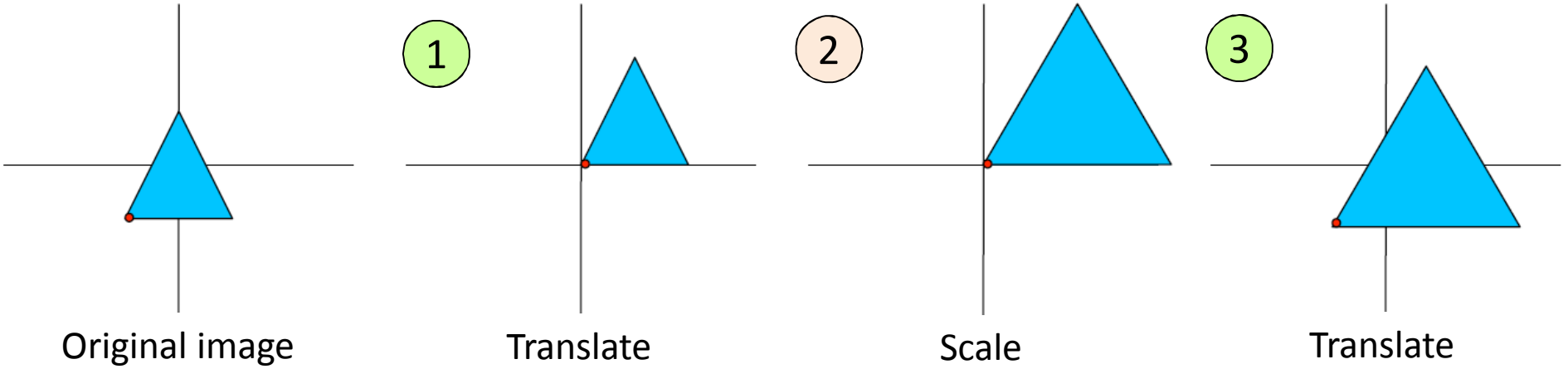
Method2 : Rotation then Translation



Original image | Rotate | Translate

Combined Geometric Transformation Operations

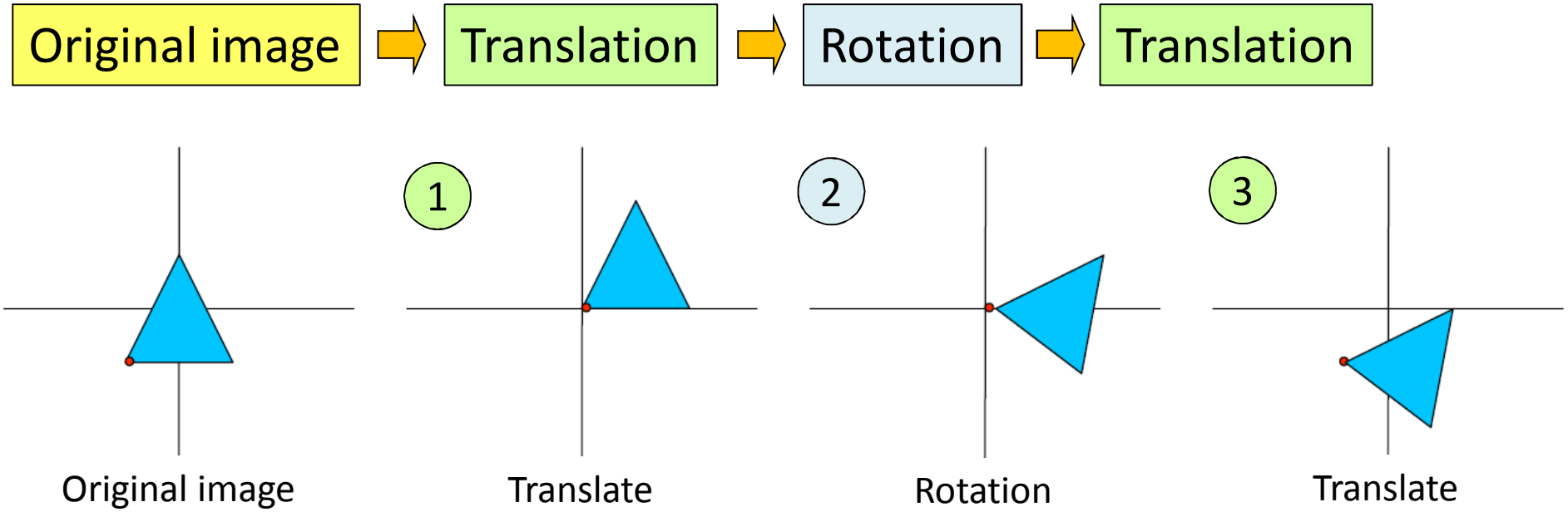
Original image → Translation → Scaling → Translation



1) Translation 2) Scaling 3) Translation

$$x' = ((x - T_x) / S_x) + T_x$$
$$y' = ((y - T_y) / S_y) + T_y$$

Combined Geometric Transformation Operations



1) Translation 2) Rotation 3) Translation :

$$x' = ((x - T_x).\cos\theta - (y - T_y).\sin\theta)) + T_x$$

$$y' = ((x - T_x).\sin\theta + (y - T_y).\cos\theta)) + T_y$$

Combined Geometric Transformation Operations

- Geometric transformation operations can be performed using Homogeneous coordinates and multiplication of 3x3 affine matrices:

Translation	$x' = x + T_x$ $y' = y + T_y$	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
Scaling	$x' = x.S_x$ $y' = y.S_y$	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
Rotation	$x' = x.\cos\theta - y.\sin\theta$ $y' = x.\sin\theta + y.\cos\theta$	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$
Shearing	$x' = x + y.Sh_x$ $y' = y + x.Sh_y$	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

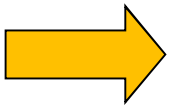
Combined Geometric Transformation Operations

Translation	$x = x' - T_x$ $y = y' - T_y$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
Scaling	$x = x'.(1/S_x)$ $y = y'.(1/S_y)$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
Rotation	$x = x'.\cos\theta + y'.\sin\theta$ $y = -x'.\sin\theta + y'.\cos\theta$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$
Shearing	$x = x' + y'.(1/Sh_x)$ $y = y' + x'.(1/Sh_y)$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$

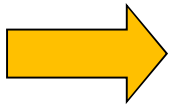
Example : Combined Geometric Operations

- Example : If we have a sequence to transform image below.

Scaling
$s_x = 0.5$
$s_y = 1$



Rotation around the center of an image, having 100x100 image size
$\theta = 45°$



Translation
$t_x = 200$
$t_y = 50$

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad Rc = \begin{bmatrix} \cos\theta & -\sin\theta & (-x_c\cos\theta + y_c\sin\theta + x_c) \\ \sin\theta & \cos\theta & (-x_c\sin\theta - y_c\cos\theta + y_c) \\ 0 & 0 & 1 \end{bmatrix} \quad Tr = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$Map = Tr \bullet Rc \bullet S = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Example : Combined Geometric Operations

$$S = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Rc = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & \left(-50\cos 45^\circ + 50\sin 45^\circ + 50\right) \\ \sin 45^\circ & \cos 45^\circ & \left(-50\sin 45^\circ - 50\cos 45^\circ + 50\right) \\ 0 & 0 & 1 \end{bmatrix}$$

$$Rc = \begin{bmatrix} 0.707 & -0.707 & 50 \\ 0.707 & 0.707 & -20.7 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Tr = \begin{bmatrix} 1 & 0 & 200 \\ 0 & 1 & 50 \\ 0 & 0 & 1 \end{bmatrix}$$

Example : Combined Geometric Operations

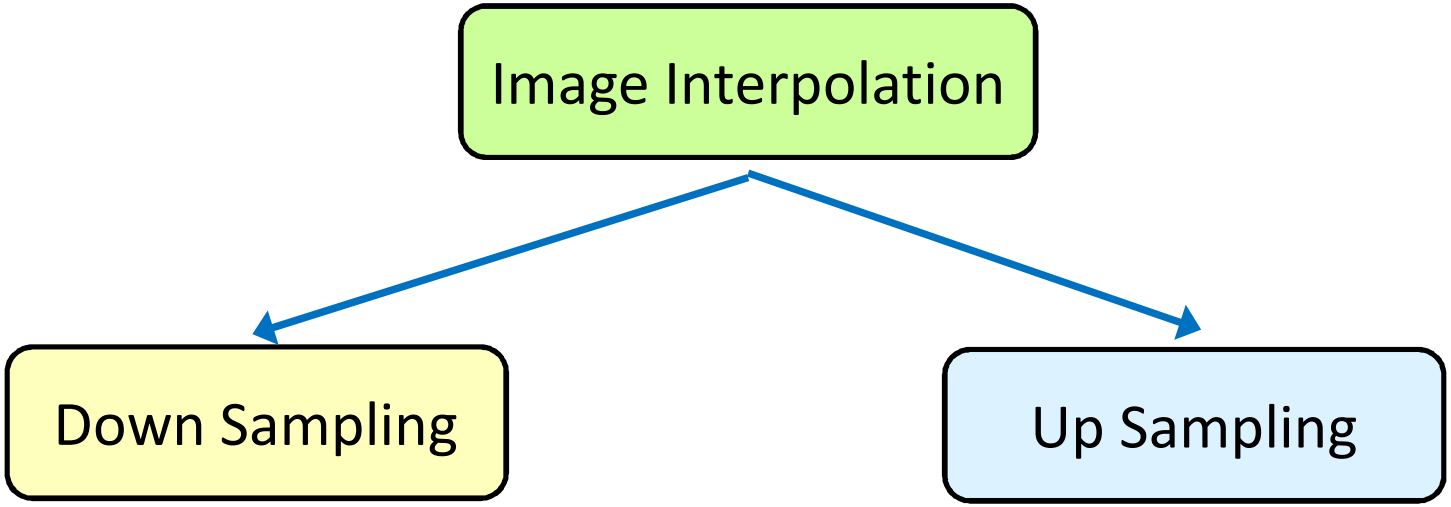
$$Rc \bullet S = \begin{bmatrix} 0.707 & -0.707 & 50 \\ 0.707 & 0.707 & -20.7 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.3535 & -0.707 & 50 \\ 0.3535 & 0.707 & -20.7 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Map = Tr \bullet Rc \bullet S = \begin{bmatrix} 1 & 0 & 200 \\ 0 & 1 & 50 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.3535 & -0.707 & 50 \\ 0.3535 & 0.707 & -20.7 \\ 0 & 0 & 1 \end{bmatrix}$$

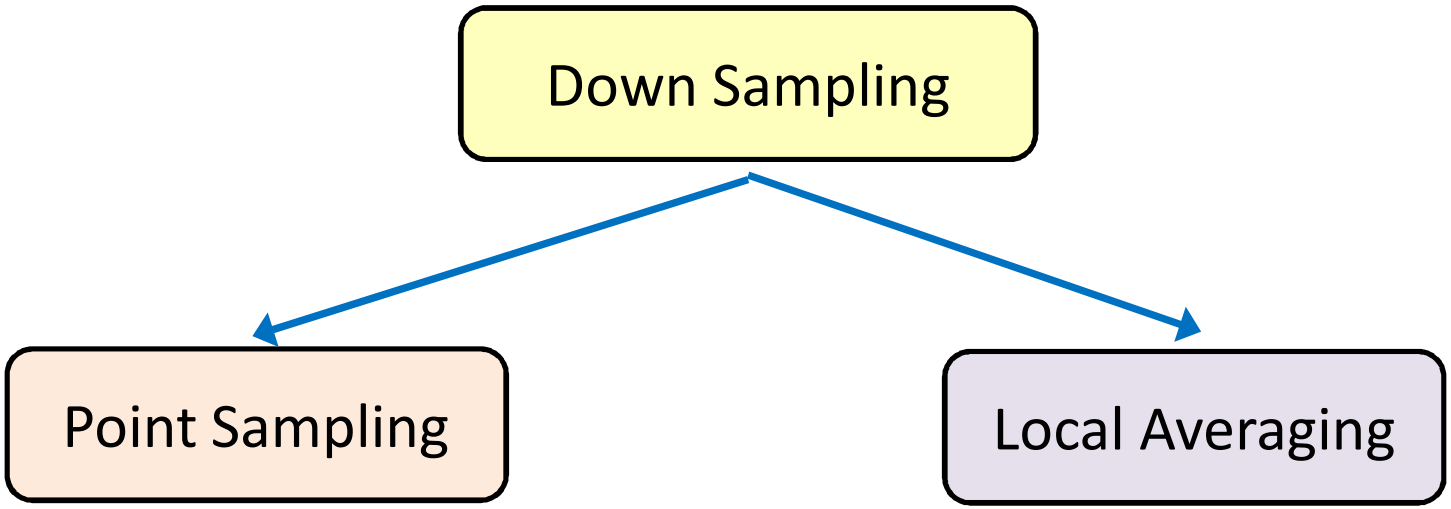
$$Map = Tr \bullet Rc \bullet S = \begin{bmatrix} 0.3535 & -0.707 & 250 \\ 0.3535 & 0.707 & 29.3 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Tr \bullet Rc \bullet S \neq S \bullet Rc \bullet Tr$$

Image Interpolation

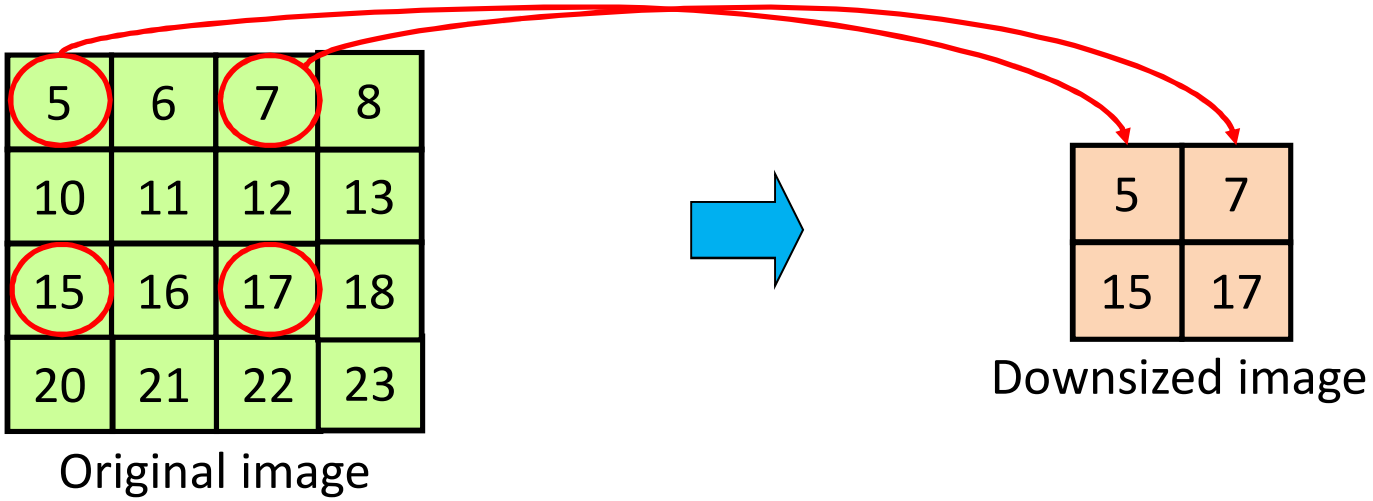


Down Sampling



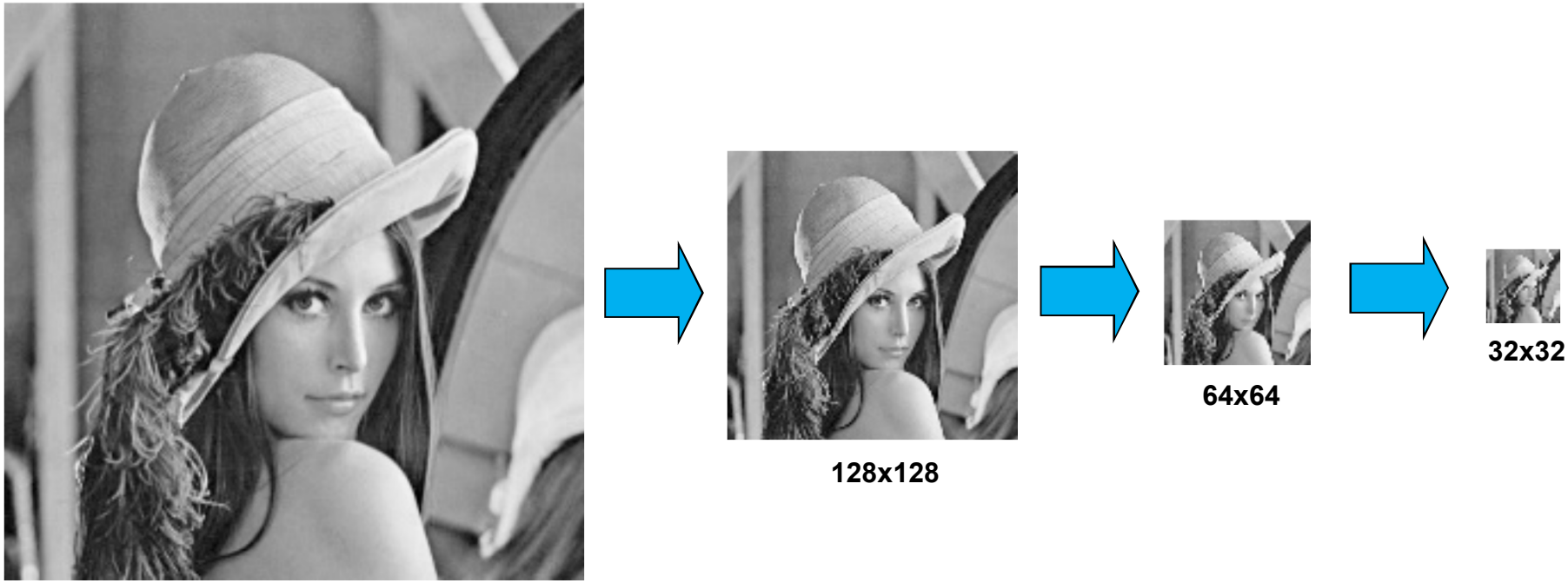
Point Sampling

- Point sampling : It used to quickly interpolate an original image down to half the size in x and y.



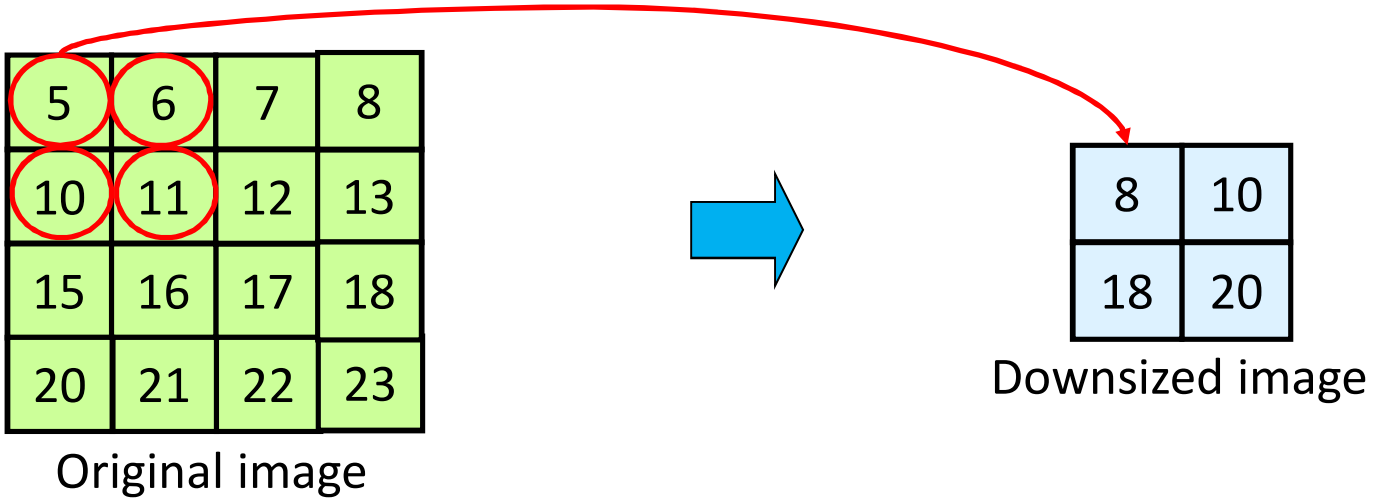
- Advantage : fast and easy to implement
- Disadvantage : possible loss of data

Example : Point Sampling



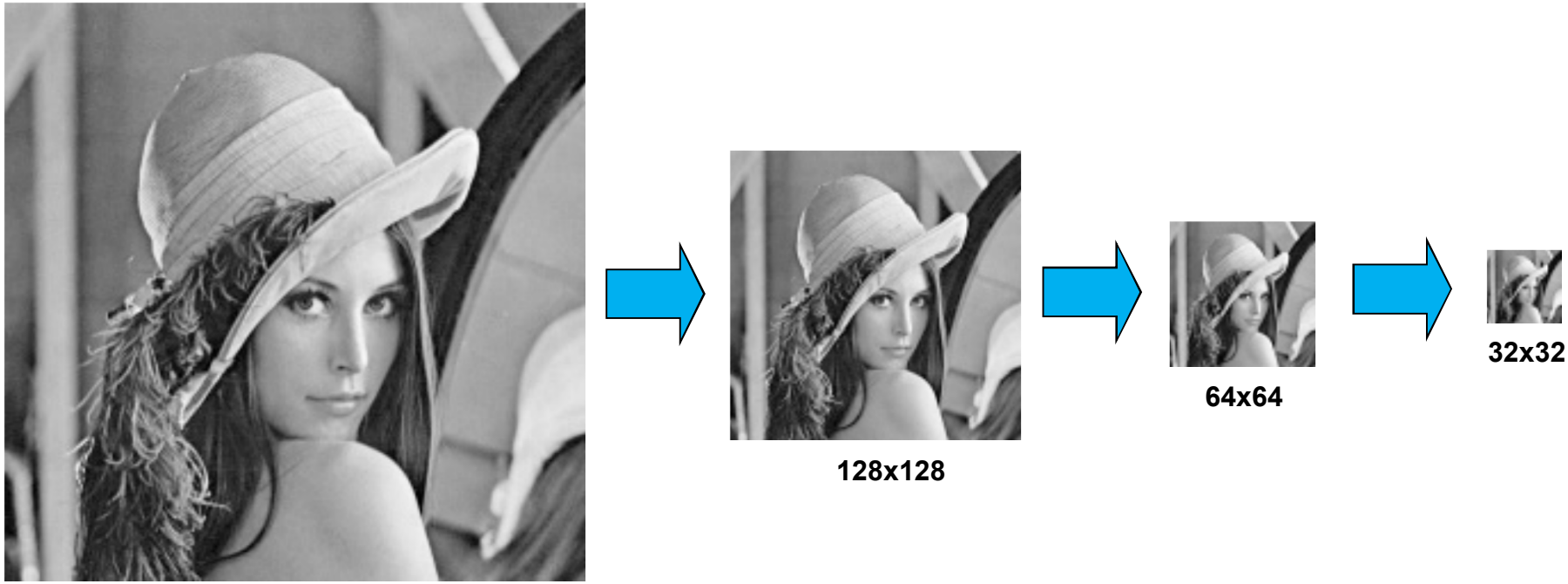
Local Averaging

- Local averaging : use average of 2x2 neighborhood to calculate output pixels.

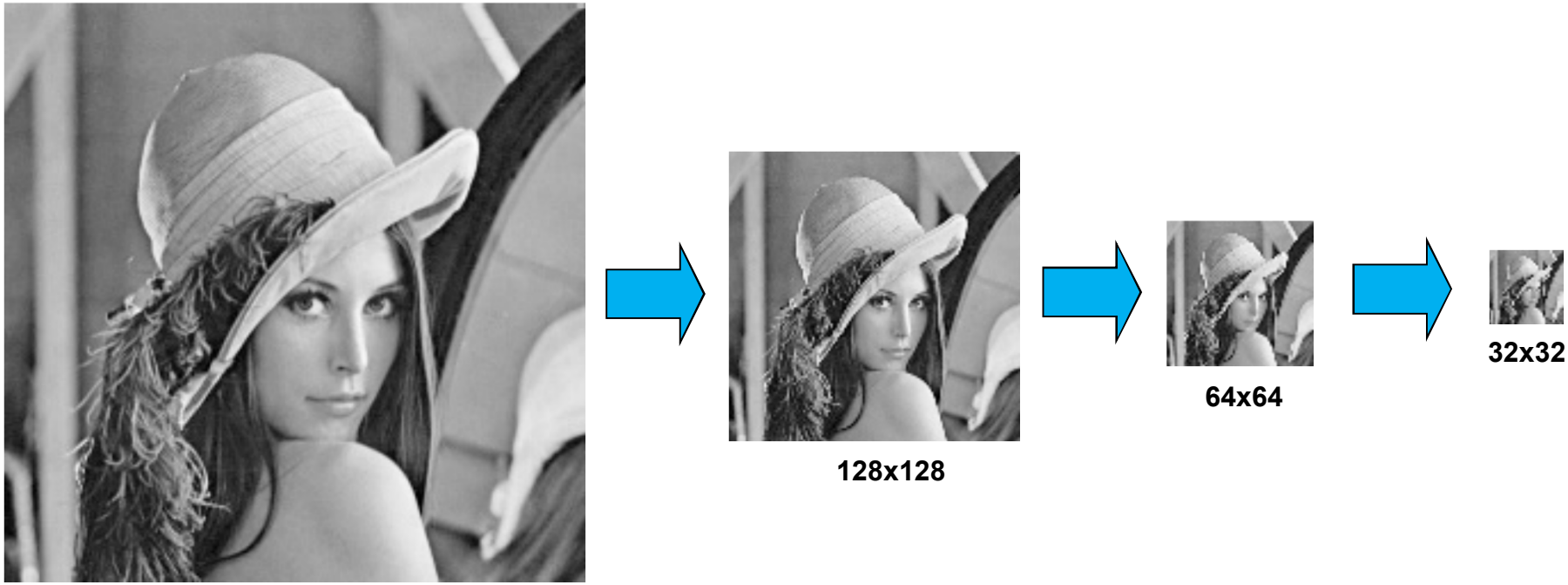


- Advantage : uses all image pixels
- Disadvantage : slightly slower

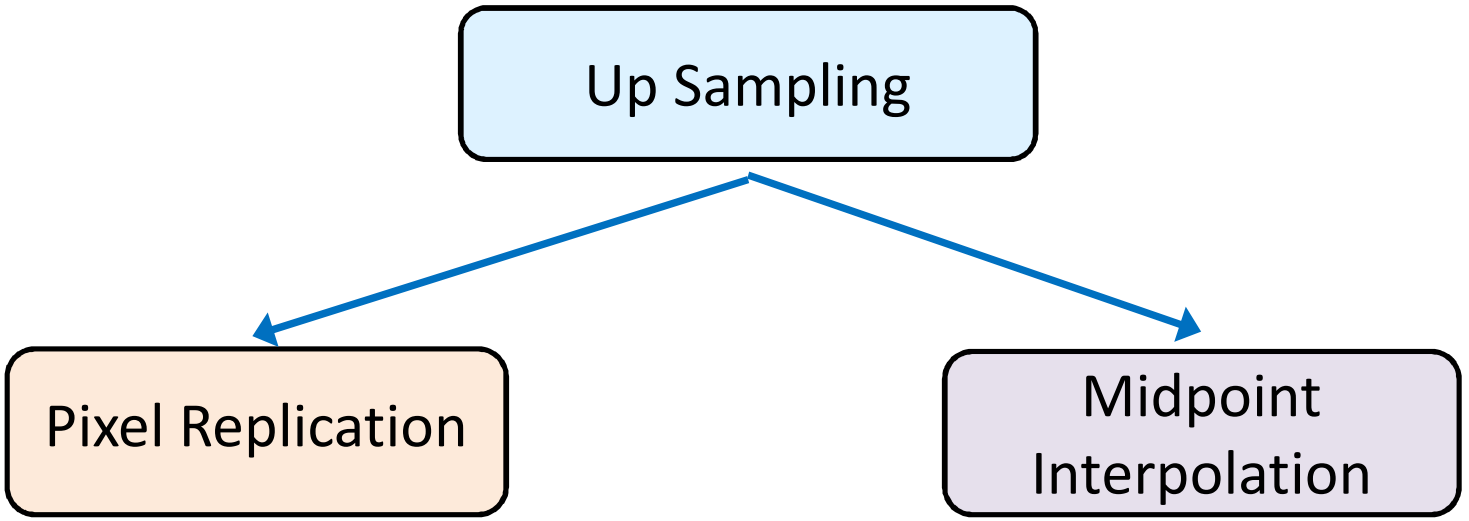
Example : Local Averaging



Example : Point Sampling

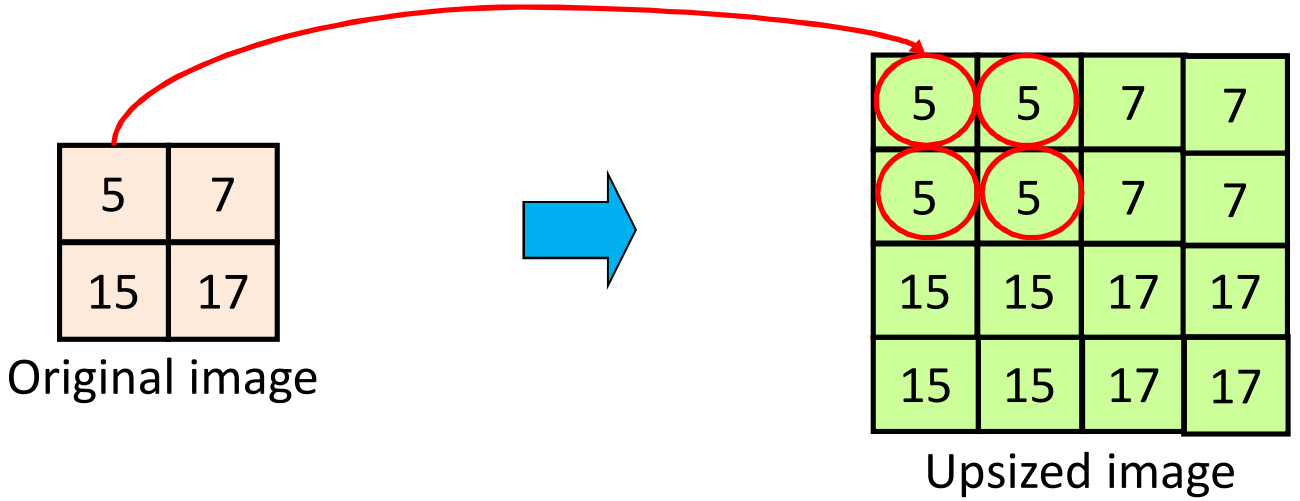


Up Sampling



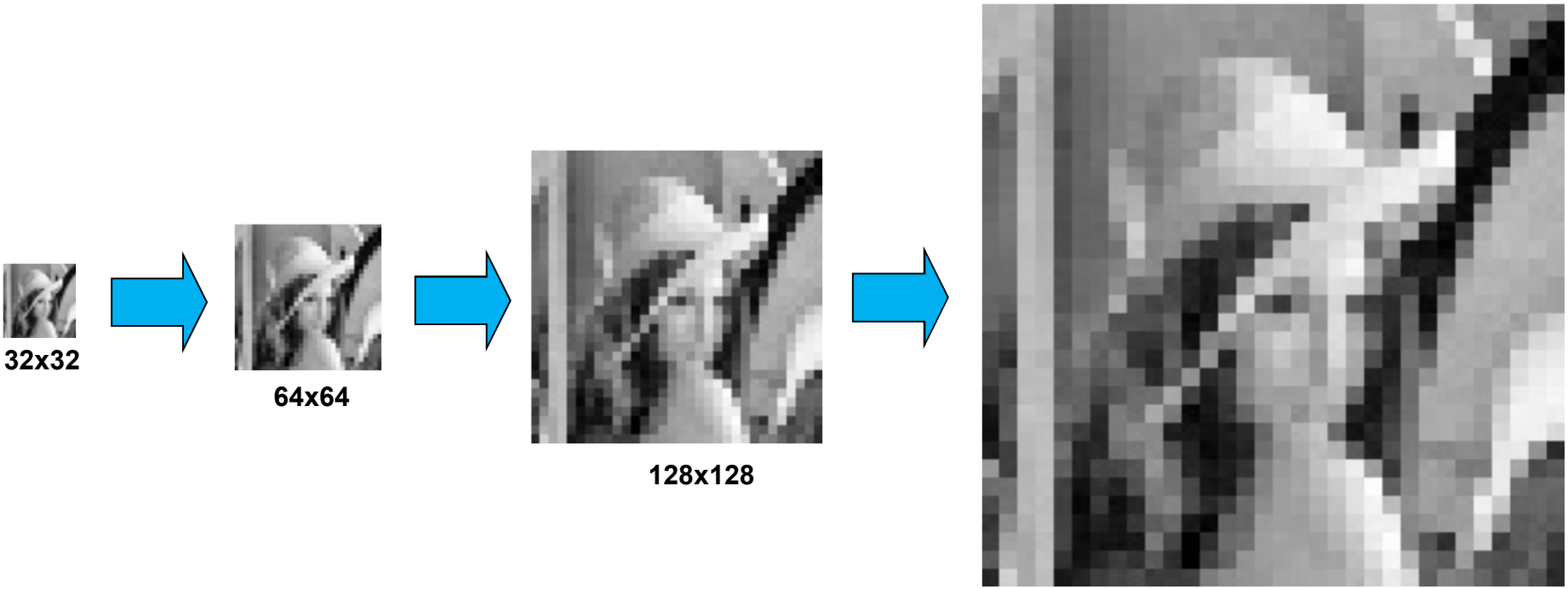
Pixel Replication

- Pixel replication : repeat the pixel values to double the size of an image.



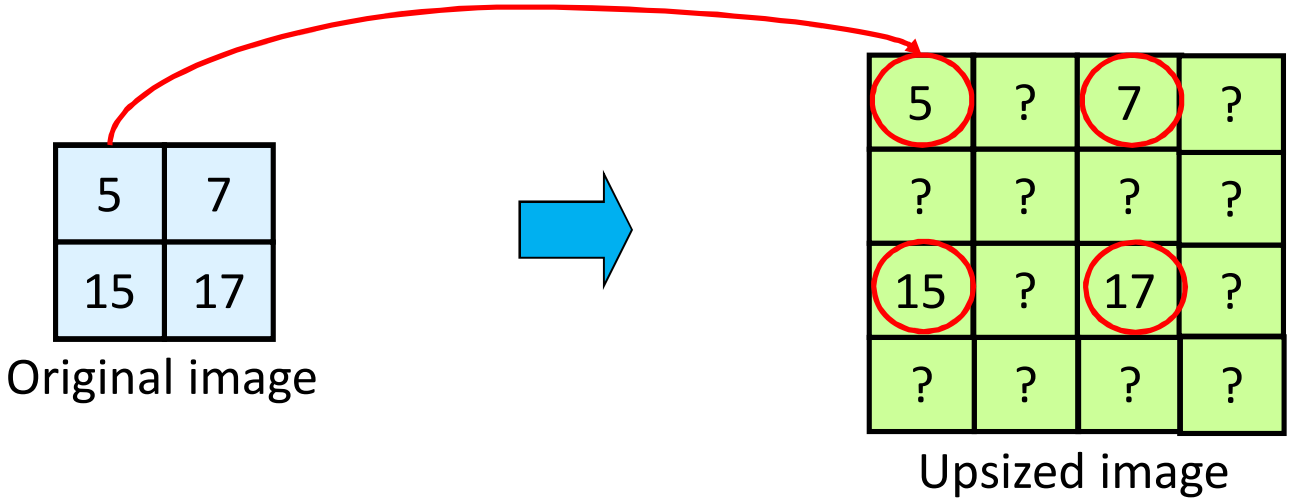
- Advantage : very easy to implement
- Disadvantage : produces undesired checkerboard pattern (aliasing) effect

Example : Pixel Replication



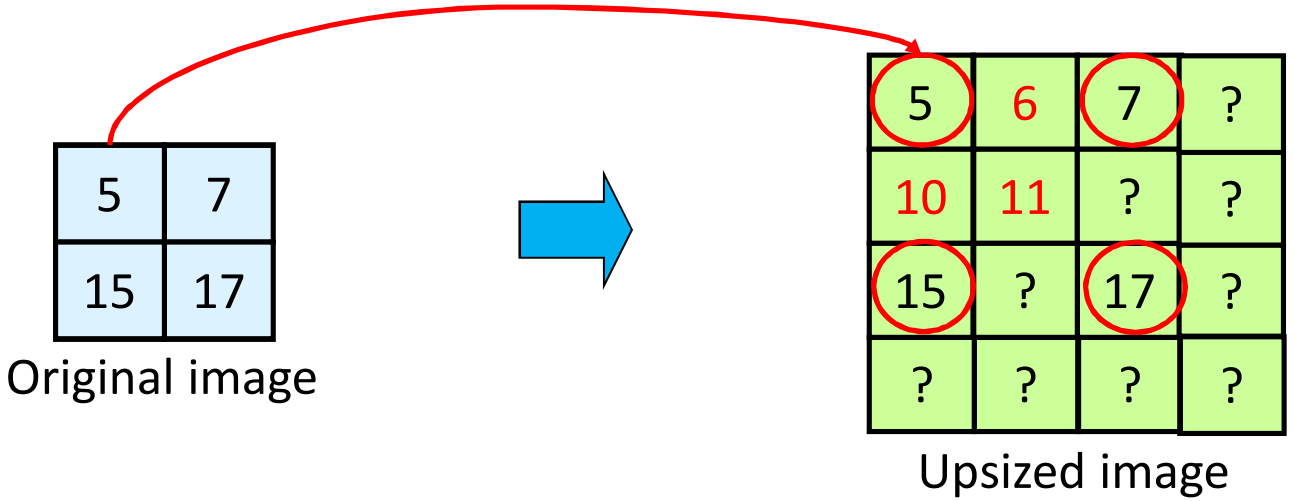
Midpoint Interpolation

- Midpoint interpolation use values half way between adjacent pixels to fill in output pixels.



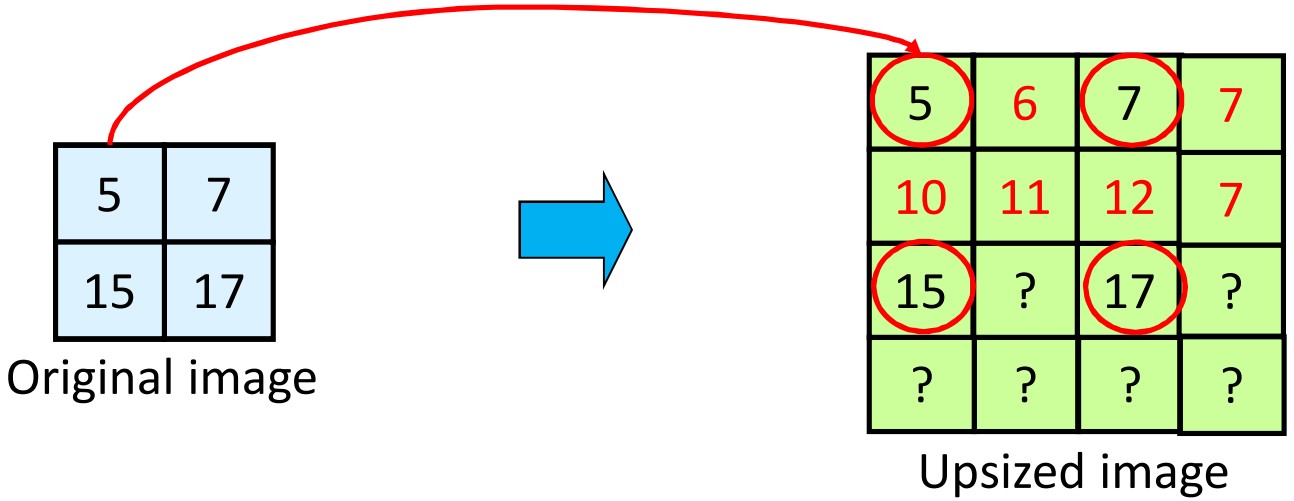
Midpoint Interpolation

- Midpoint interpolation – use values half way between adjacent pixels to fill in output pixels.



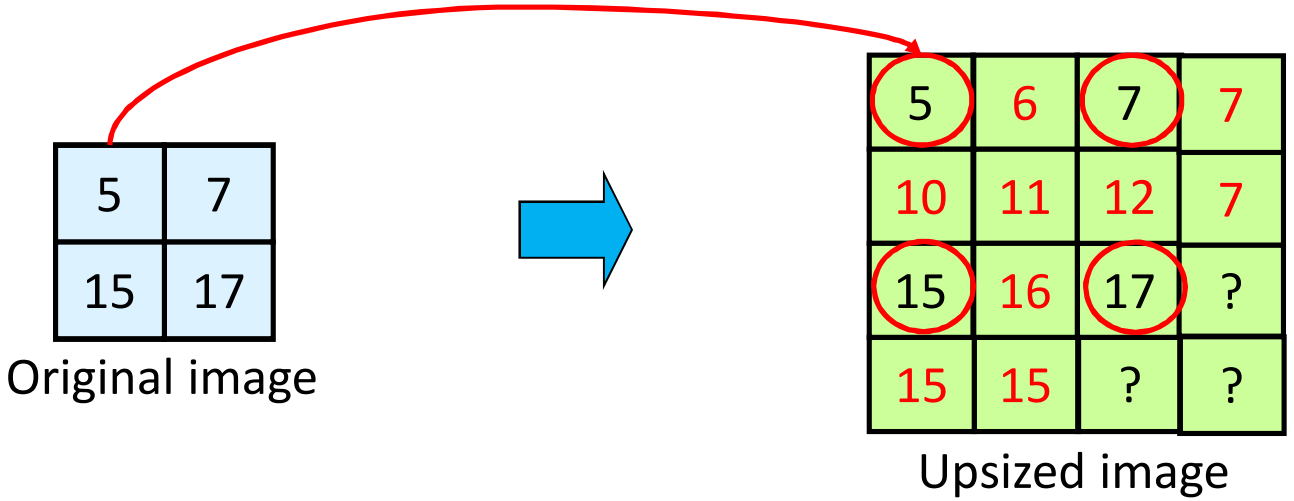
Midpoint Interpolation

- Midpoint interpolation – use values half way between adjacent pixels to fill in output pixels.



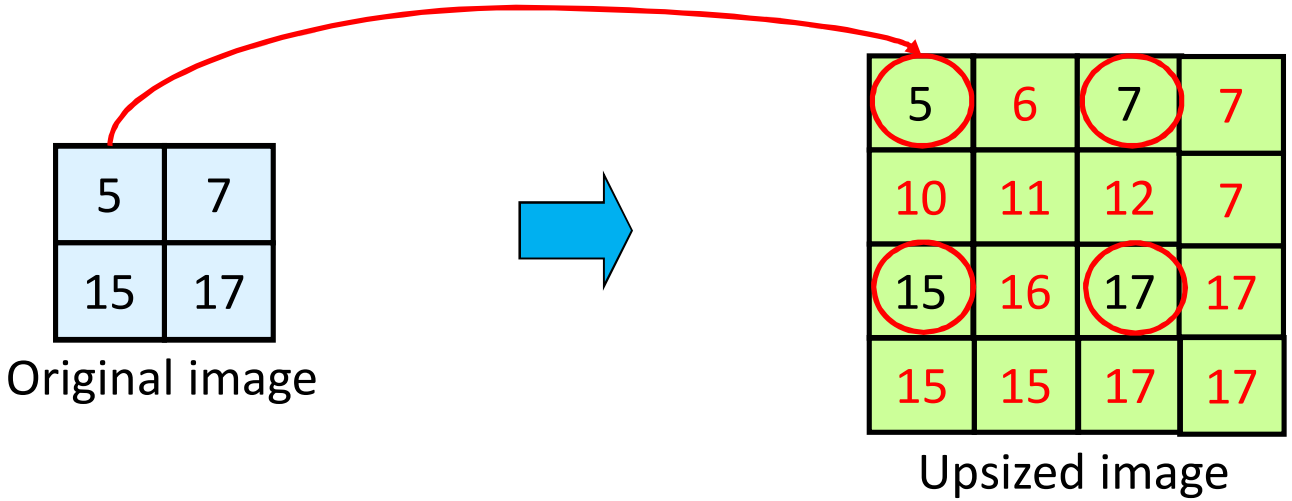
Midpoint Interpolation

- Midpoint interpolation – use values half way between adjacent pixels to fill in output pixels.



Midpoint Interpolation

- Midpoint interpolation – use values half way between adjacent pixels to fill in output pixels.



- Advantage : results in smoother output image
- Disadvantages :
  - uses more additions and divisions
  - must handle last row and column as a special case (typically using pixel replication)

Example : Midpoint Interpolation

Example : Pixel Replication

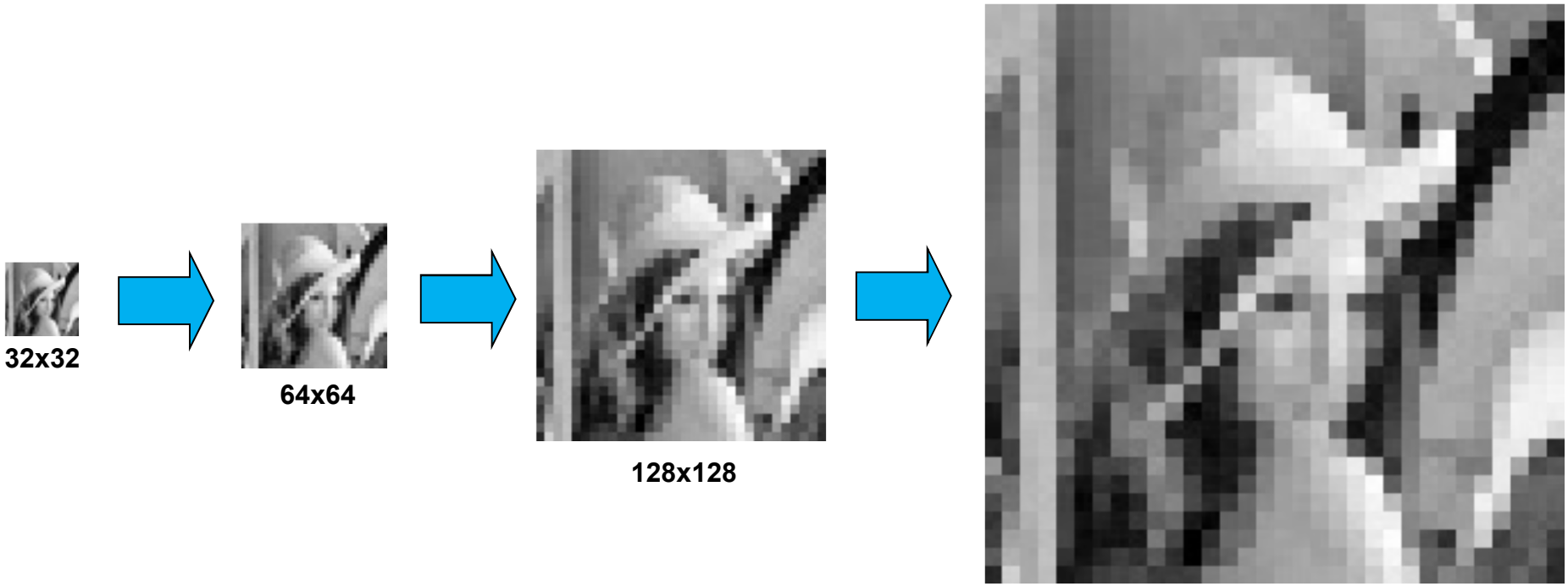
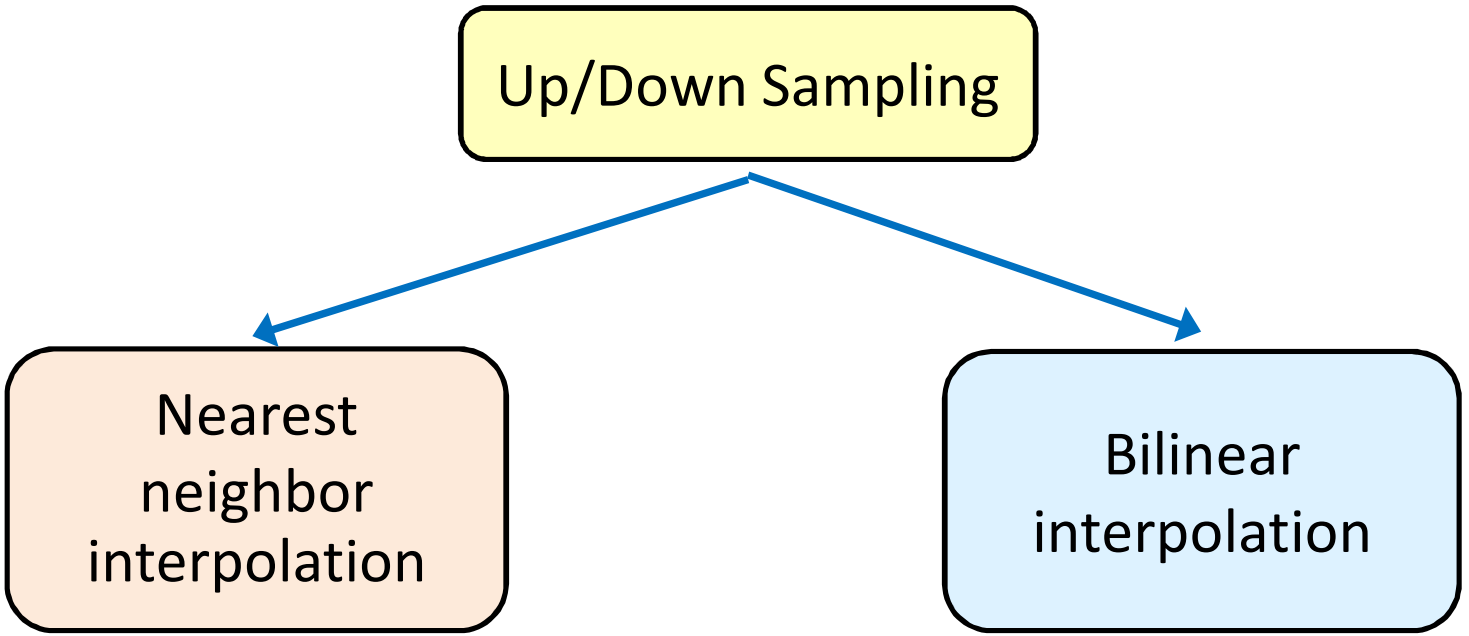


Image Interpolation



Nearest Neighbor Interpolation

- Nearest neighbor interpolation use rounding to get closest integer value. It can be used to interpolate images up/down to any size.

- Advantage : fast and easy to implement
- Disadvantage : data loss or block artifacts

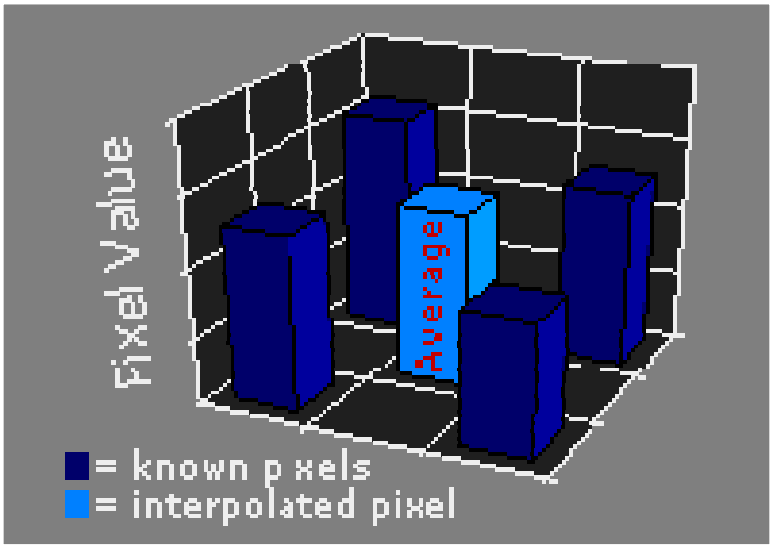
Example : Nearest Neighbor Interpolation

Resized image
180x300

Resized image
300x180

Bilinear Interpolation

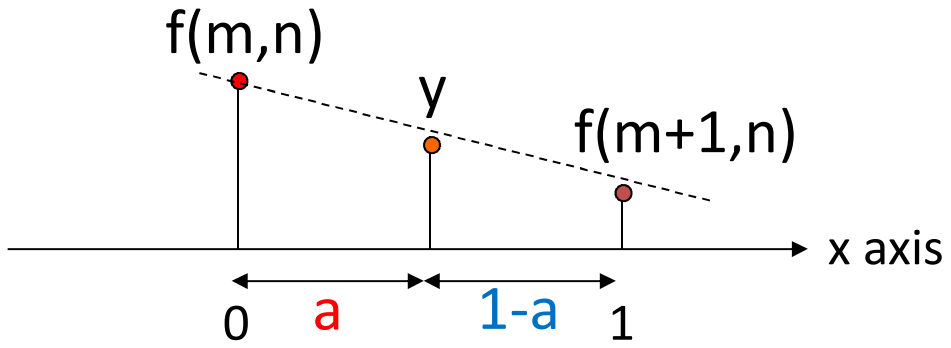
- Bilinear interpolation use distance weighted average of adjacent pixels to obtain output pixel value. It can be used to interpolate images up/down to any size.



- Advantage : more accurate than nearest neighbor
- Disadvantage : slower and more complex to implement

Bilinear Interpolation

- Method of linear: the closer to a pixel, the higher weight is assigned



$$y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$$

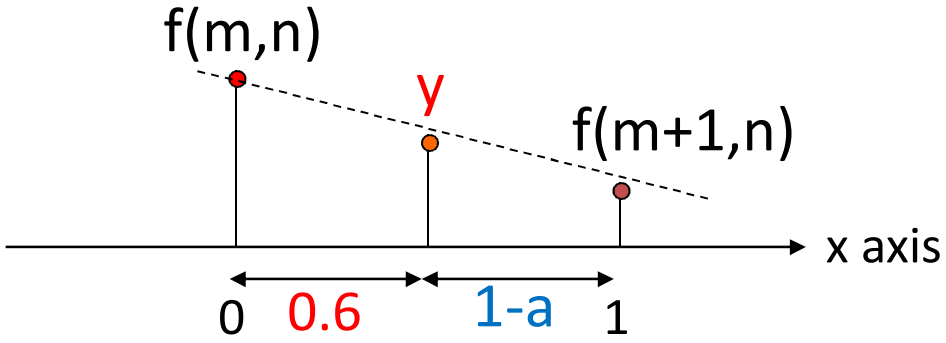
$$y - f(m,n) = \frac{(f(m+1,n) - f(m,n))}{(1-0)}(a-0)$$

$$y = (1-a) \times f(m,n) + a \times f(m+1,n)$$

- When a=0.5, we can use the average of two values, 0<a<1

Example : Linear Interpolation

- Define f(m,n)=10 and f(m+1,n)=40



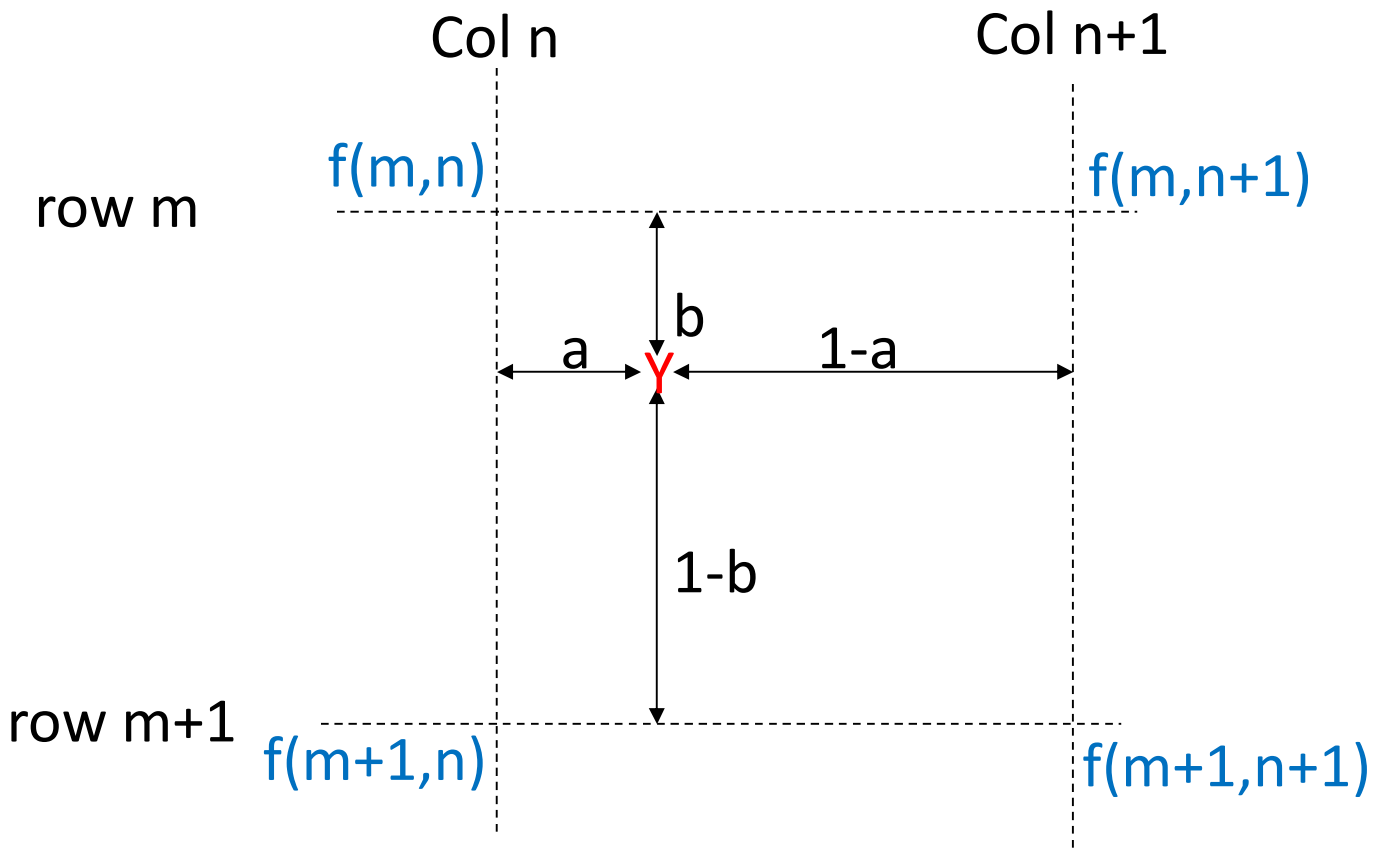
$$y = (1-a) \times f(m,n) + a \times f(m+1,n)$$

$$y = (0.4 \times 10) + (0.6 \times 40)$$

$$y = 28$$

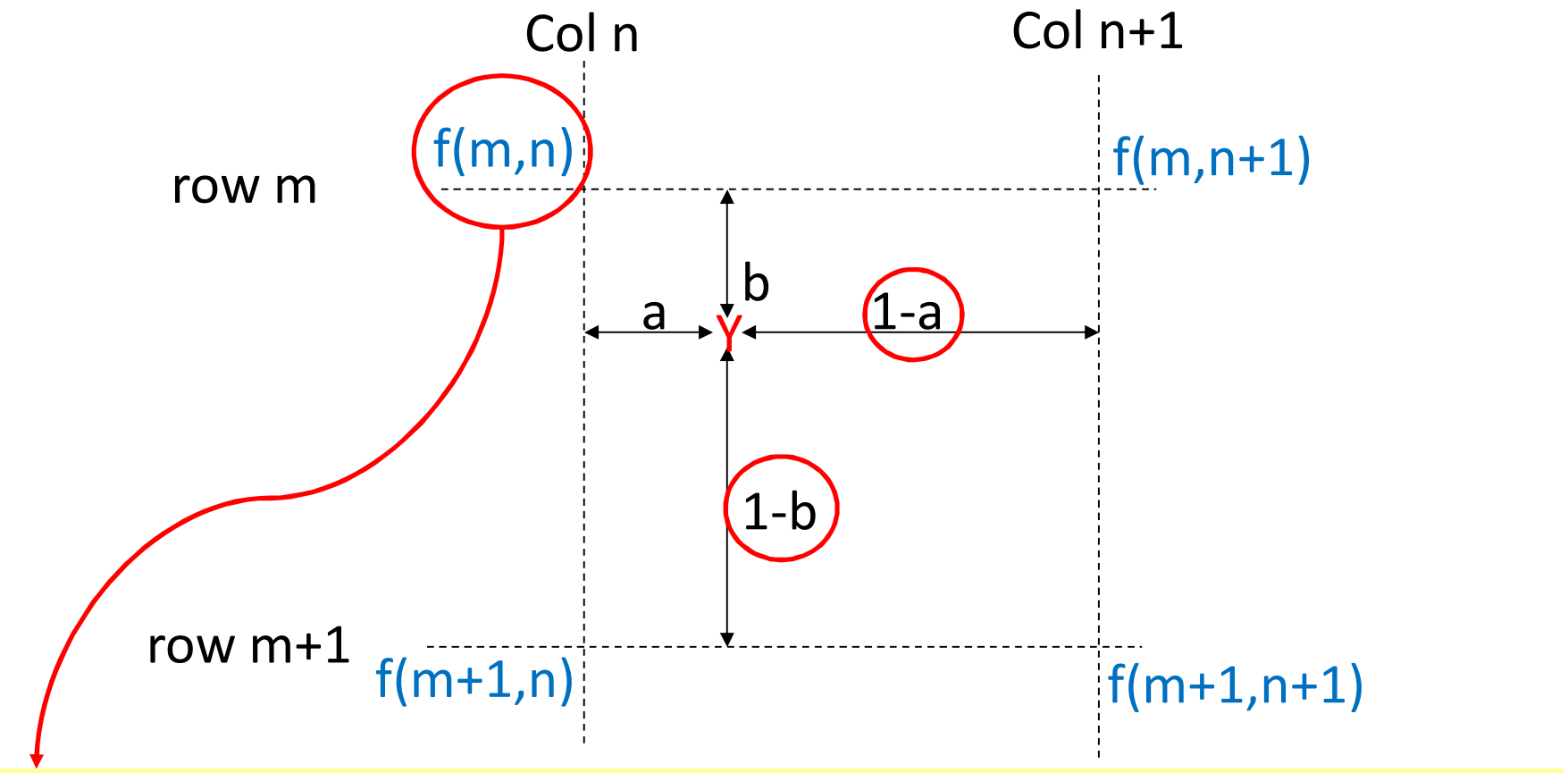
Bilinear Interpolation

- What is the interpolated value at Y?



Bilinear Interpolation

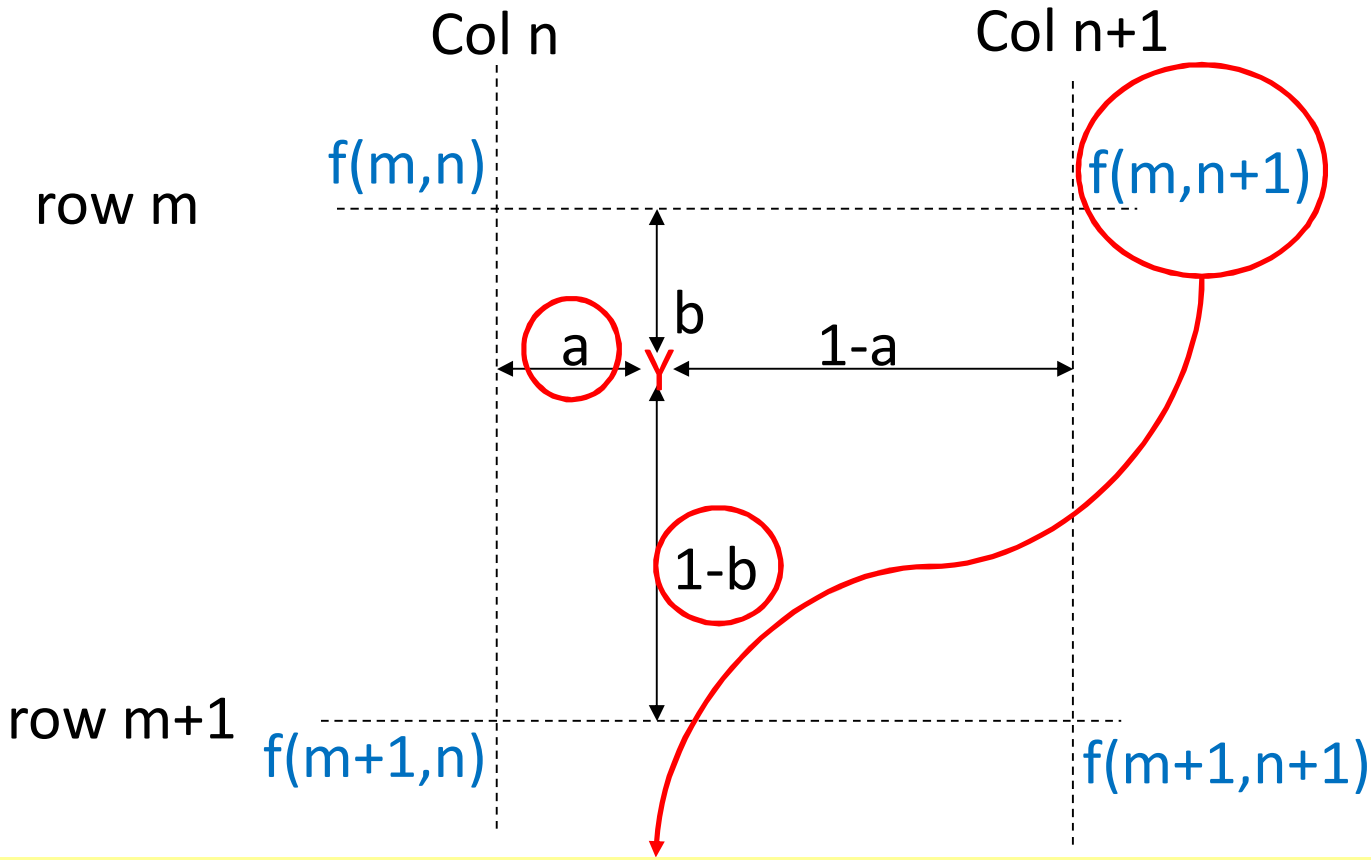
- What is the interpolated value at Y?



Ans: (1-a)×(1-b)×f(m,n)

Bilinear Interpolation

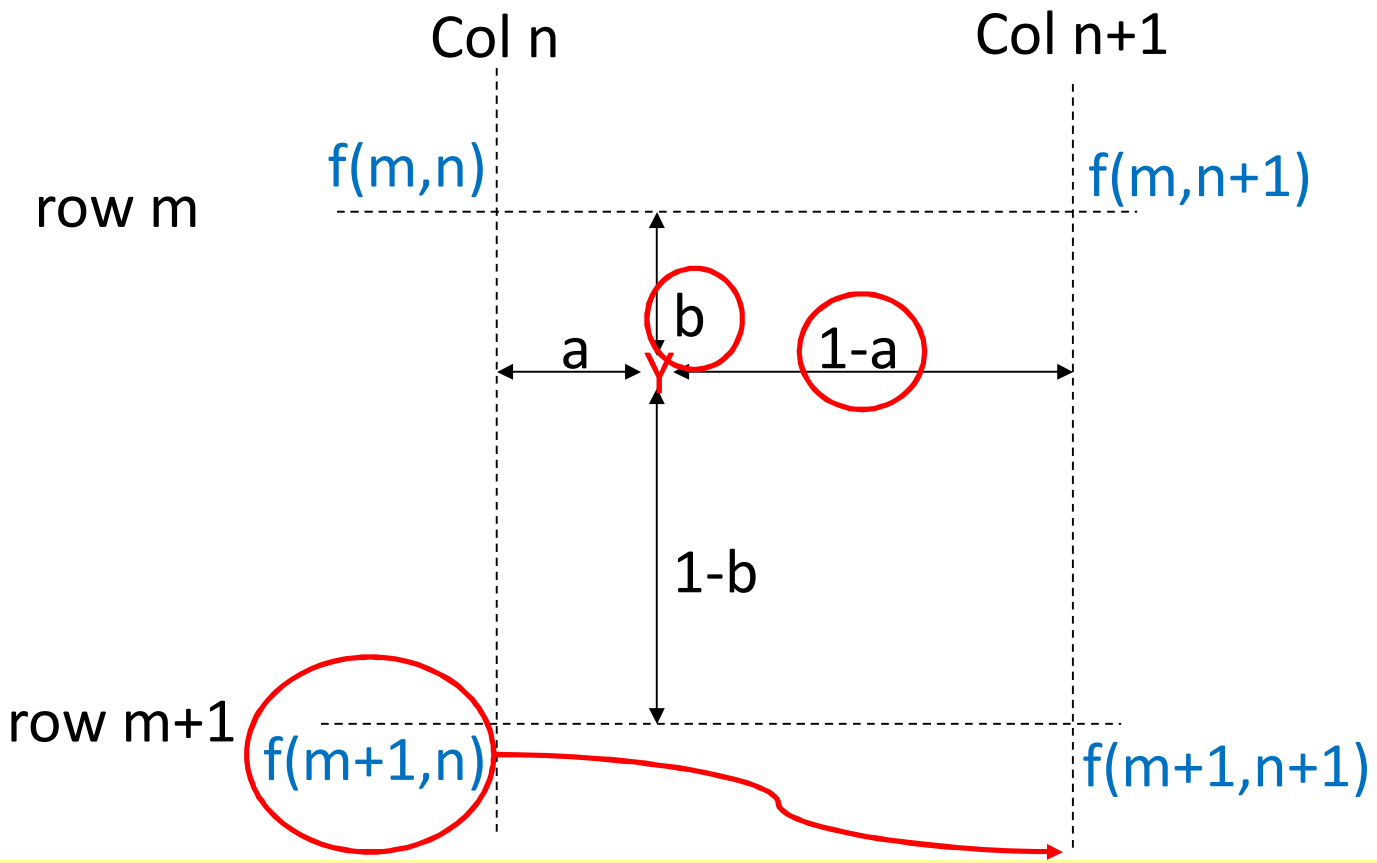
- What is the interpolated value at Y?



Ans: (1-a)×(1-b)×f(m,n)+a×(1-b)×f(m,n+1)

Bilinear Interpolation

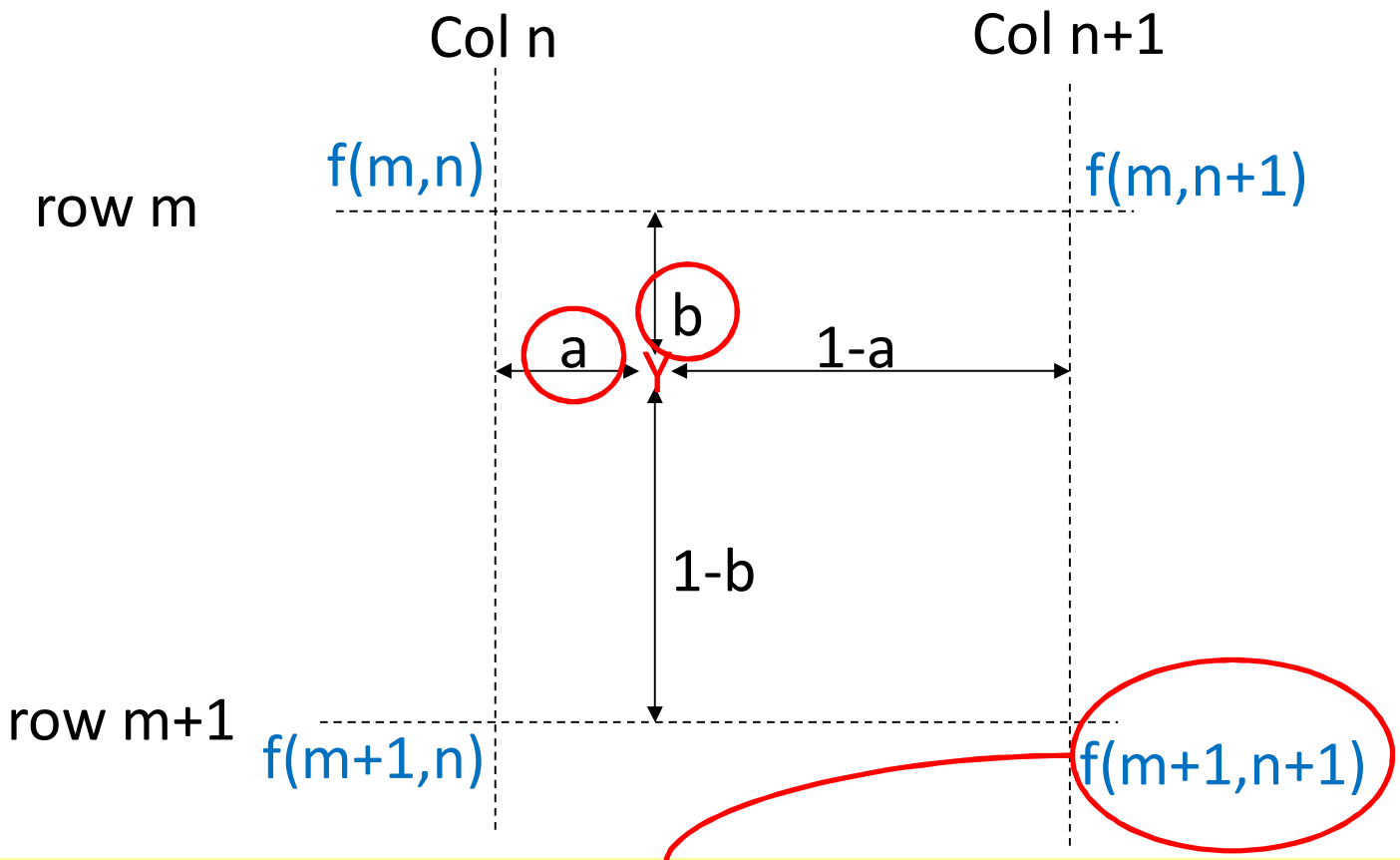
- What is the interpolated value at Y?



Ans: (1-a)×(1-b)×f(m,n)+a×(1-b)×f(m,n+1)+(1-a)×b×f(m+1,n)

Bilinear Interpolation

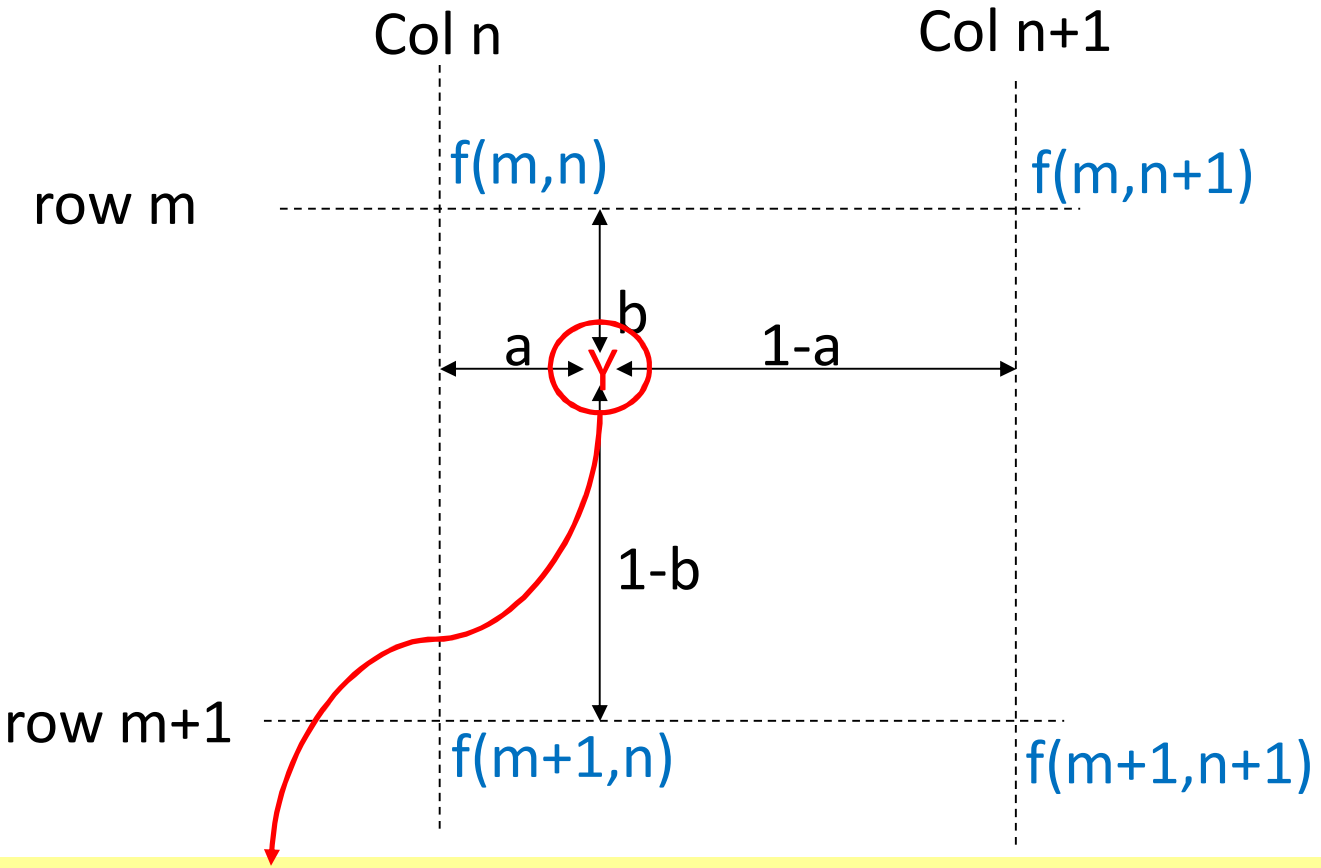
- What is the interpolated value at Y?



Ans: (1-a)×(1-b)×f(m,n)+a×(1-b)×f(m,n+1)+(1-a)×b×f(m+1,n)
+a×b×f(m+1,n+1)

Bilinear Interpolation

- What is the interpolated value at Y?



Ans: (1-a)×(1-b)×f(m,n)+a×(1-b)×f(m,n+1)+(1-a)×b×f(m+1,n)
+a×b×f(m+1,n+1)

Example : Bilinear Interpolation

Resized image
180x300

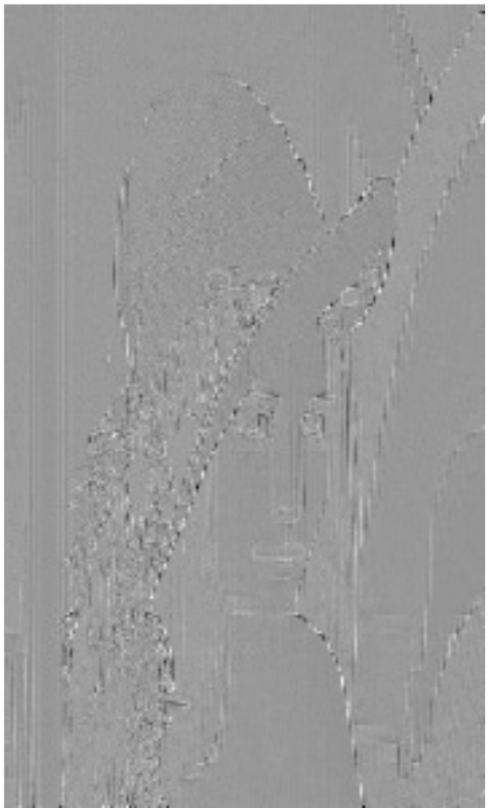
Resized image
300x180

Example : Nearest Neighbor Interpolation

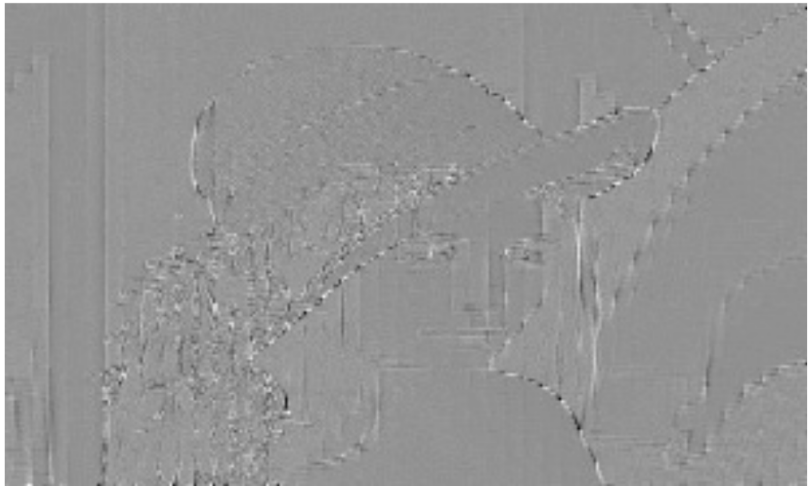
Resized image
180x300

Resized image
300x180

Compared Bilinear with Nearest Neighbor



Resized image
180x300



Resized image
300x180

Compared Bilinear with Nearest Neighbor

- Nearest neighbor interpolation:
  - fastest processing
  - produces undesired checkerboard pattern (aliasing) effect
- Bilinear interpolation:
  - smoother looking images than nearest neighbor.
  - has an anti-aliasing effect, therefore less aliasing effect than nearest neighbor.

Thanks for your attention