Geometric Transformations

- Modify the arrangement of pixels based on some geometric transformation.

• Translation

$$r' = r + t_r$$
$$c' = c + t_c$$

• Scaling

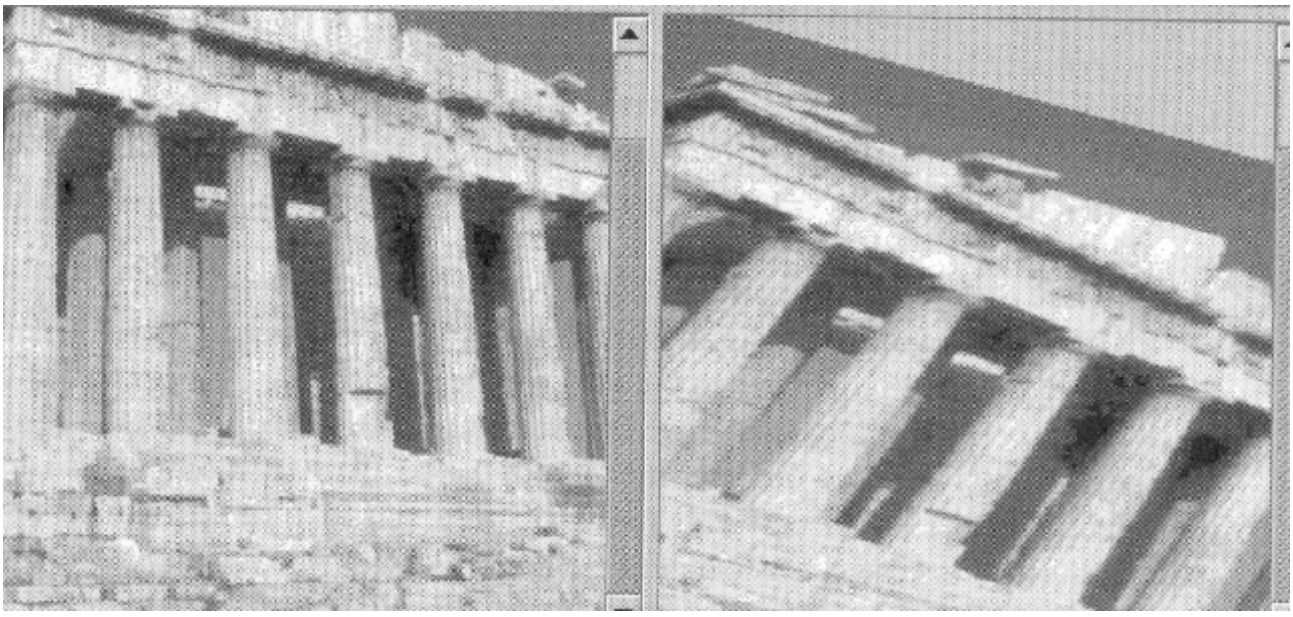
$$r' = s_r r$$
$$c' = s_c c$$

• Rotation

$$r' = r_0 + (r - r_0)cos(\theta) - (c - c_0)sin(\theta)$$
$$c' = c_0 + (r - r_0)sin(\theta) + (c - c_0)cos(\theta)$$

• Affine Transformation

$$r' = a_{11}r + a_{12}c + b_1$$
$$c' = a_{21}r + a_{22}c + b_2$$

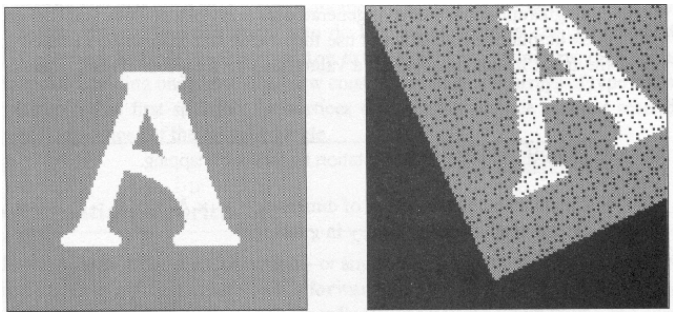


• Some Practical Problems

(1) Transformed pixel coordinates might not lie within the bounds of the image.

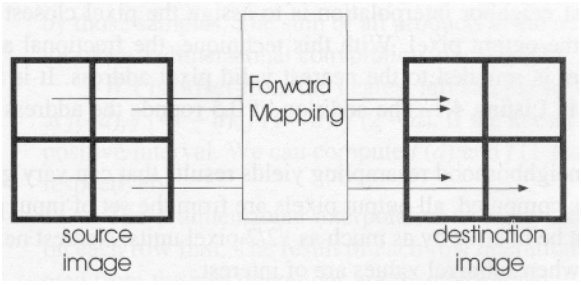
(2) Transformed pixel coordinates can be non-integer.

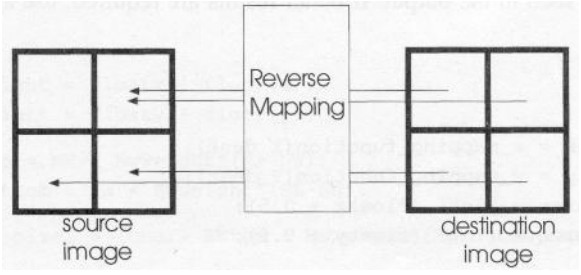
(3) There might be no pixels in the input image that map to certain pixel locations in the transformed image (one-to-one correspondence can be lost).



• (3)--> Forward vs Inverse Mapping

- To guarantee that a value is generated for every pixel in the output image, we must consider each output pixel in turn and use the *inverse* mapping to determine the position in the input image.

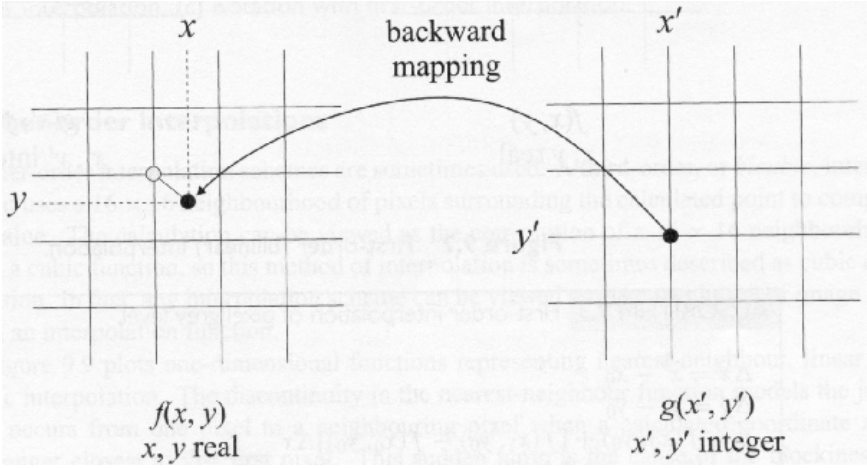




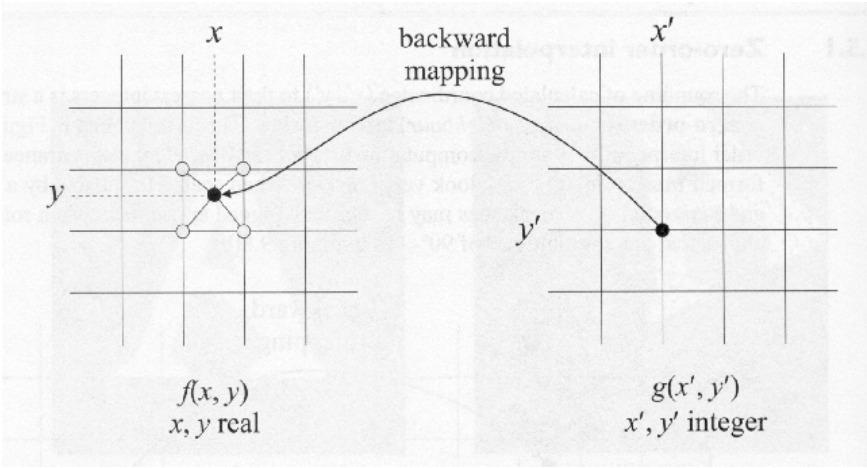
• (2)--> Image Interpolation

- Interpolation is the process of generating integer coordinates for a transformed pixel by examining its surrounding pixels.

Zero-order interpolation (or nearest-neighbor)



First-order interpolation



- Higher-order interpolation schemes are more sophisticated but also more time consuming (see notes).