

INTRODUCTION TO DIGITAL IMAGE PROCESSING

— IMAGE TRANSFORM

Xiaohui Yuan

Department of Computer Science and Engineering
University of North Texas
xiaohui.yuan@unt.edu

Image Translation

- The translation maps each pixel in an input image into a new position in the output image
 - The contents of the image remain the same.
- Given the displacements $(\Delta x, \Delta y)$, the coordinates of pixel (x_0, y_0) is transformed to (x_1, y_1) as follows:
$$\begin{array}{l} x_1 = x_0 + \Delta x \\ y_1 = y_0 + \Delta y \end{array} \quad OR \quad \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$
- Note that the pixel value is not changed in the transformation.
- All pixels in an image are translated in the same amount



Image Rotation

- Under rotation, pixel (x_0, y_0) in an image is relocated to a new position (x_1, y_1) in the output image by an angle θ about the origin as follows:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \quad \text{OR} \quad \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

- For triangle ABP, we have $\cos(\phi) = x/r$ and $\sin(\phi) = y/r$
- For triangle ABP', we have $\cos(\theta + \phi) = x'/r$ and $\sin(\theta + \phi) = y'/r$
- Hence, we have

$$x = r \cos(\phi), y = r \sin(\phi)$$

$$x' = r \cos(\theta + \phi) = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$$

$$y' = r \sin(\theta + \phi) = r \cos(\phi) \sin(\theta) + r \sin(\phi) \cos(\theta)$$

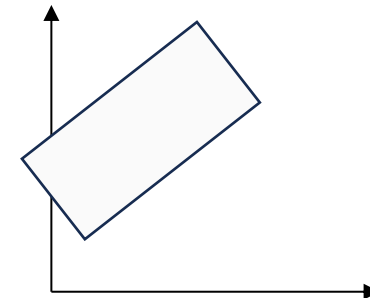
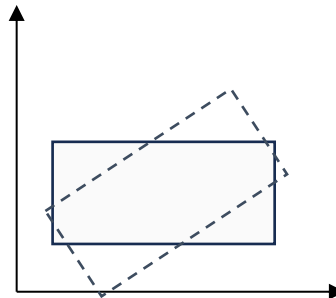
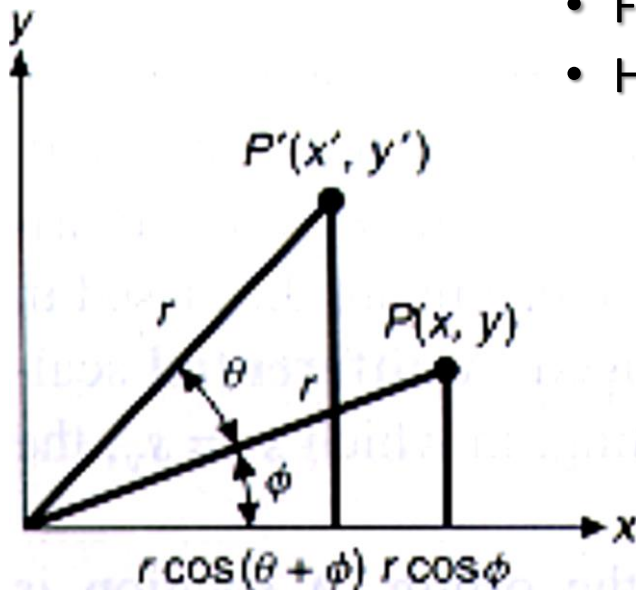


Image Scaling

- Image scaling changes the size of the object by multiplying the coordinates of the points by scaling factors.
- With scaling, a pixel (x_0, y_0) in an image is relocated to a new position (x_1, y_1) in the output image by displacing it through a user-specified scaling factor s as follows:

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = s \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

- A non-uniform scaling can be achieved with

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

- The gaps (pixels without a value) between the pixels in the output image need to be filled in with interpolation.

homogeneous coordinates

- Add one additional coordinate w , i.e.,

$$(x', y') \rightarrow (x, y, w), \quad w \neq 0$$

- Recover (x', y') by homogenizing (x, y, w) as follows:

$$x' = \frac{x}{w}, \quad y' = \frac{y}{w}$$

- So, we have $x = x'w$, $y = y'w$,

$$(x', y') \rightarrow (x'w, y'w, w)$$

- Translation in homogeneous coordinates:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x' + dx \\ y' + dy \\ 1 \end{bmatrix}$$

homogeneous coordinates

- Scaling

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x' \\ s_y y' \\ 1 \end{bmatrix}$$

- Rotation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x' \cos(\theta) - y' \sin(\theta) \\ x' \sin(\theta) + y' \cos(\theta) \\ 1 \end{bmatrix}$$

Successive Transformation

- Successive translations

$$\begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & dx' \\ 0 & 1 & dy' \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx + dx' \\ 0 & 1 & dy + dy' \\ 0 & 0 & 1 \end{bmatrix}$$

- Successive scaling

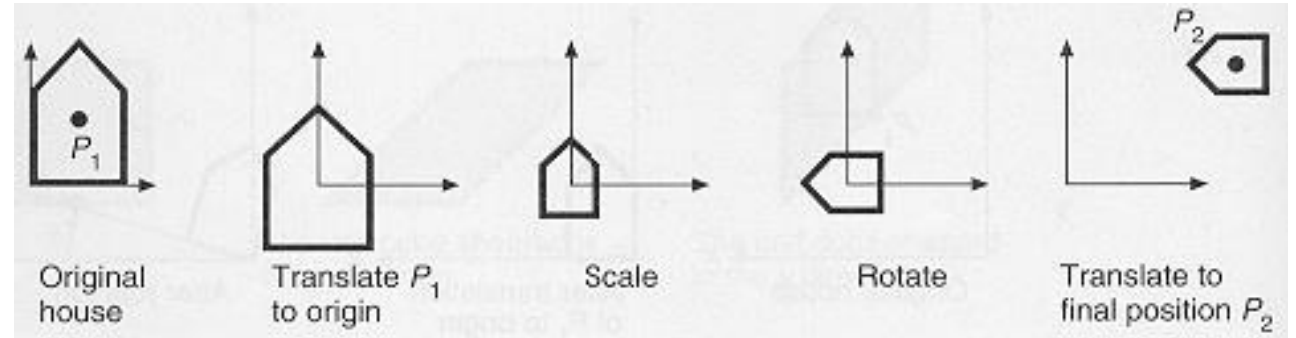
$$\begin{bmatrix} s_{x'} & 0 & 0 \\ 0 & s_{y'} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x s_{x'} & 0 & 0 \\ 0 & s_y s_{y'} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Successive rotations

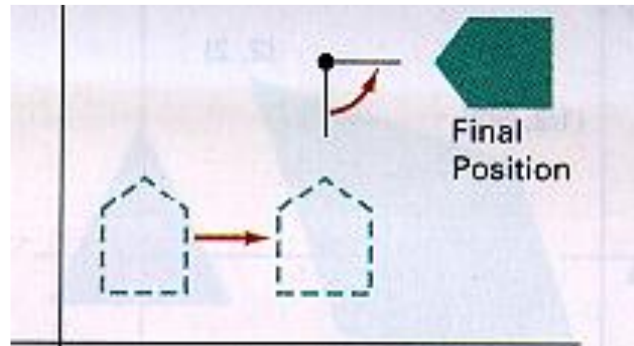
$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta + \phi) & -\sin(\theta + \phi) & 0 \\ \sin(\theta + \phi) & \cos(\theta + \phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composition of Transformations

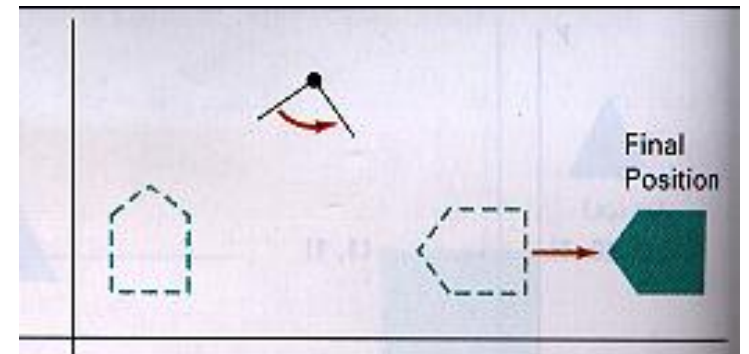
- The transformation matrices of a series of transformations can be concatenated into a single transformation matrix.
- Example:
 - Translate to origin
 - Perform scaling and rotation
 - Translate to location P_2
- Important: **preserve the order** of transformations!



translation +
rotation



rotation +
translation



General form of transformation

- The general form of transformation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \overset{\text{Rotation and scale}}{a_{11} & a_{12}} & \overset{\text{translation}}{a_{13}} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

- Representing a sequence of transformations as a single transformation matrix

$$\begin{aligned} x &= a_{11}x' + a_{12}y' + a_{13} \\ y &= a_{21}x' + a_{22}y' + a_{23} \end{aligned}$$

Special cases of transformations

- Rigid transformations: involve translation and rotation
 - Preserve angles and lengths
- Similarity transformations: involve rotation, translation, and scaling
 - Preserve angles but not lengths
- Affine transformations: involve translation, rotation, scaling, and shear
 - Preserve parallelism of lines but not lengths or angles.
 - Shearing along x-axis: $x = x' + ay, y = y'$ or $\begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
 - Shearing along y-axis: $x = x', y = y' + bx$ or $\begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Image Registration

- Give two images I_1 and I_2 , image registration finds transformation functions f and g such that after applying these functions to image I_1 the transformed image matches I_2 without error

$$I_2(x, y) = g(I_1(f(x, y)))$$

where f is a 2D spatial transformation function and g is a 1D intensity transformation function



Registration Approaches

- Control point-based registration
 - Fast
 - Any registration problem
 - User operation is needed
 - Accuracy relies on experience
- Content-based registration
 - Automatic process
 - Performance is consistent
 - High computational expense

