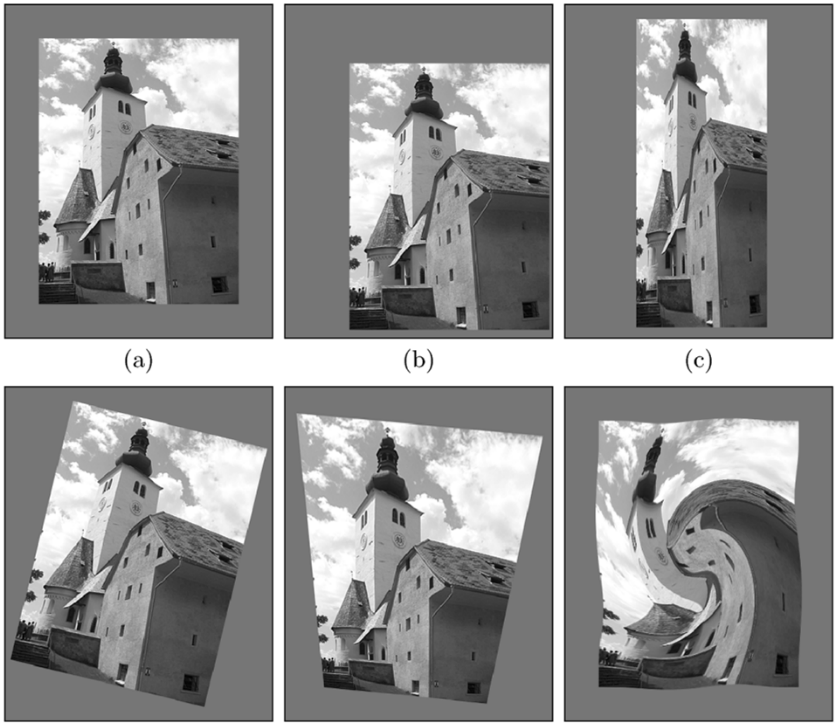
# EIE4512 - Digital Image Processing
# Geometric Operations

香 港 中 文 大 學（深 圳）
The Chinese University of Hong Kong, Shenzhen

Zhen Li
lizhen@cuhk.edu.cn
School of Science and Engineering
The Chinese University of Hong Kong, Shen Zhen
April 4, 2019

# Geometric Operations

- Filters, point operations change intensity
- Pixel position (and geometry) unchanged
- Geometric operations: change image geometry
- **Examples:** translating, rotating, scaling an image



(a) (b) (c) (d) (e) (f)

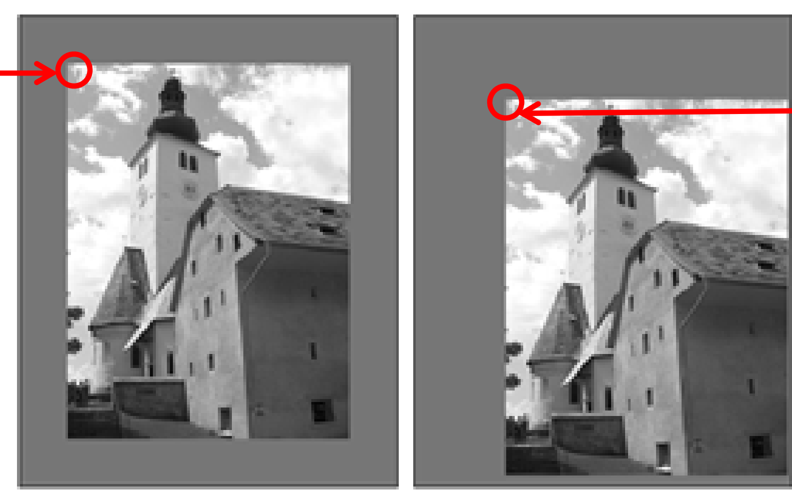
**Examples of Geometric operations**

# Geometric Operations

- Example applications of geometric operations:
  - Zooming images, windows to arbitrary size
  - Computer graphics: deform textures and map to arbitrary surfaces
- **Definition:** Geometric operation transforms image *I* to new image *I'* by modifying **coordinates of image pixels**

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

- Intensity value originally at (x,y) moved to new position (x',y')




**Example: Translation geometric operation moves value at (x,y) to (x + $d_x$, y + $d_y$)**

# Geometric Operations

- Since image coordinates can only be discrete values, some transformations may yield (x’,y’) that’s not discrete
- **Solution:** interpolate nearby values

# Simple Mappings

- **Translation:** (shift) by a vector $(d_x, d_y)$

$$T_x : x' = x + d_x \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$
$$T_y : y' = y + d_y \quad \text{or}$$

- **Scaling:** (contracting or stretching) along x or y axis by a factor $s_x$ or $s_y$

$$T_x : x' = s_x \cdot x \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$
$$T_y : y' = s_y \cdot y \quad \text{or}$$

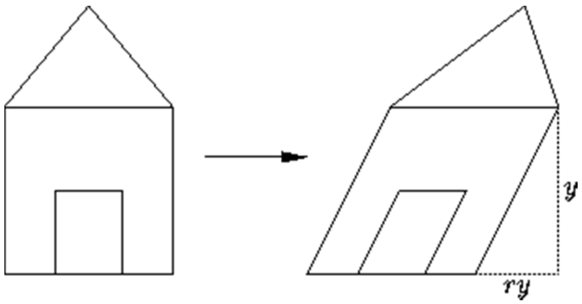
# Simple Mappings

- **Shearing:** along x and y axis by factor $b_x$ and $b_y$

$$T_x : x' = x + b_x \cdot y$$
$$T_y : y' = y + b_y \cdot x$$

or

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & b_x \\ b_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$




- **Rotation:** the image by an angle $\alpha$

$$T_x : x' = x \cdot \cos\alpha - y \cdot \sin\alpha$$
$$T_y : y' = x \cdot \sin\alpha + y \cdot \cos\alpha$$

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

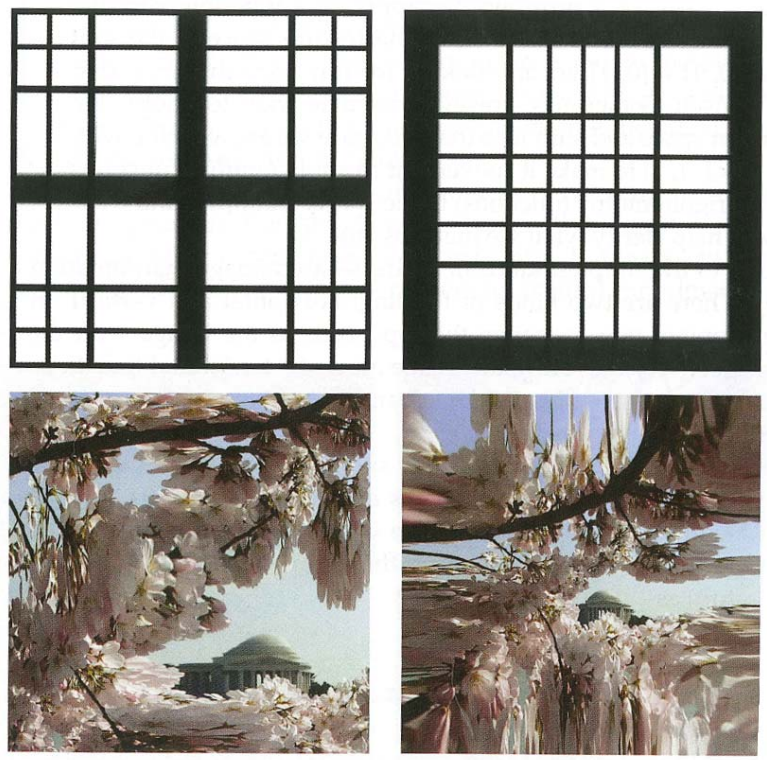
# Image Flipping & Rotation by 90 degrees

- We can achieve 90,180 degree rotation easily
- Basic idea: Look up a **transformed pixel address** instead of the current one
- To flip an image upside down:
  - At pixel location $xy$, look up the color at location $x\ (1 - y)$
- For horizontal flip:
  - At pixel location $xy$, look up $(1 - x)\ y$
- Rotating an image 90 degrees counterclockwise:
  - At pixel location $xy$, look up $(y, 1 - x)$

# Image Flipping, Rotation and Warping

- **Image warping:** we can use a function to select which pixel somewhere else in the image to look up
- For example: apply function on both texel coordinates (x, y)

$$x = x + y * \sin(\pi * x)$$



# Homogeneous Coordinates

- Notation useful for converting scaling, translation, rotating into point-matrix multiplication
- To convert ordinary coordinates into homogeneous coordinates

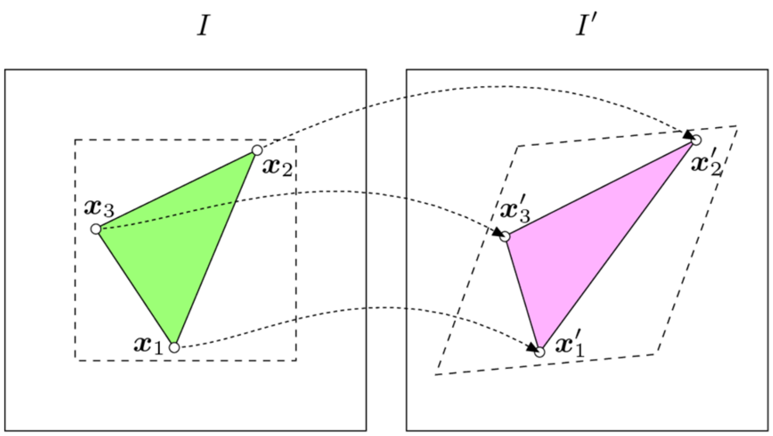
$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{converts to} \quad \hat{\boldsymbol{x}} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ h \end{pmatrix} = \begin{pmatrix} h\,x \\ h\,y \\ h \end{pmatrix}$$

# Affine (3-Point) Mapping

- Can use homogeneous coordinates to rewrite translation, rotation, scaling, etc as vector-matrix multiplication

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

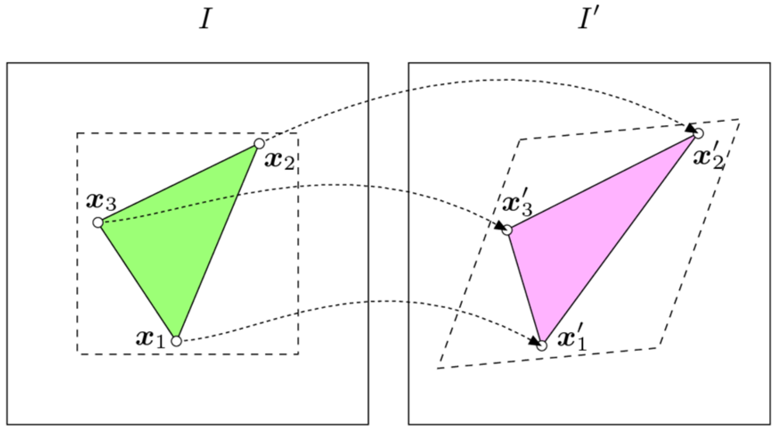
- **Affine mapping:** Can then derive values of matrix that achieve desired transformation (or combination of transformations)




- Inverse of transform matrix is **inverse mapping**

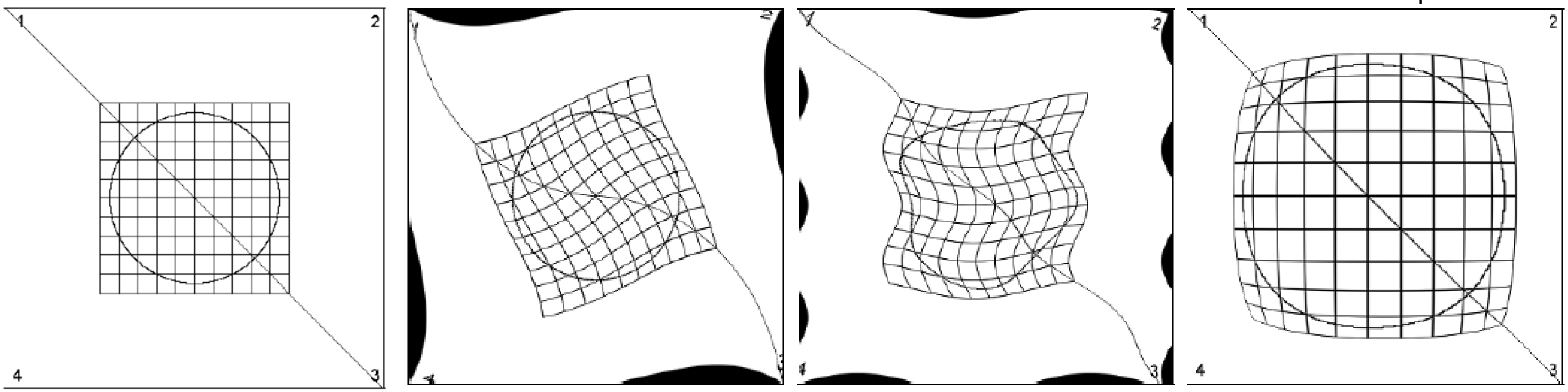
# Affine (3-Point) Mapping

- **What's so special about affine mapping?**




- Maps
  - straight lines -> straight lines,
  - triangles -> triangles
  - rectangles -> parallelograms
  - Parallel lines -> parallel lines
- Distance ratio on lines do not change

# Non-Linear Image Warps




(a) (b) (c)

(d) (e) (f)

**Original** **Twirl** **Ripple** **Spherical**

# Twirl

- **Notation:** Instead using texture colors at (x',y'), use texture colors at twirled (x,y) location
- Twirl?
  - Rotate image by angle ***α*** at center or anchor point ($x_c$,$y_c$)
  - Increasingly rotate image as radial distance ***r*** from center increases (up to $r_{max}$)
  - Image unchanged outside radial distance $r_{max}$

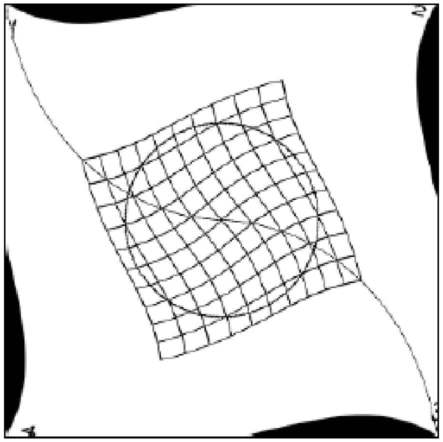
$$T_x^{-1}: \quad x = \begin{cases} x_c + r \cdot \cos(\beta) & \text{for} \quad r \leq r_{\max} \\ x' & \text{for} \quad r > r_{\max}, \end{cases}$$

$$T_y^{-1}: \quad y = \begin{cases} y_c + r \cdot \sin(\beta) & \text{for} \quad r \leq r_{\max} \\ y' & \text{for} \quad r > r_{\max}, \end{cases}$$

with

$$d_x = x' - x_c, \qquad r = \sqrt{d_x^2 + d_y^2},$$

$$d_y = y' - y_c, \qquad \beta = \mathrm{Arctan}(d_y, d_x) + \alpha \cdot \left(\tfrac{r_{\max} - r}{r_{\max}}\right).$$



(a)

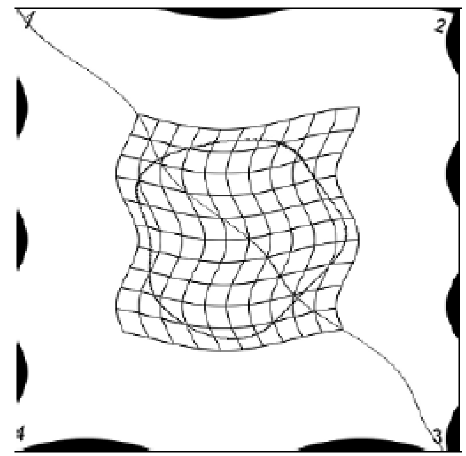
(d)

# Ripple

- Ripple causes wavelike displacement of image along both the x and y directions

$$T_x^{-1}: \quad x = x' + a_x \cdot \sin\left(\frac{2\pi \cdot y'}{\tau_x}\right),$$
$$T_y^{-1}: \quad y = y' + a_y \cdot \sin\left(\frac{2\pi \cdot x'}{\tau_y}\right).$$

- Sample values for parameters (in pixels) are
  - $\tau_x$= 120
  - $\tau_y$= 250
  - $a_x$= 10
  - $a_y$= 15



(b)

(e)

# Spherical Transformation

- Imitates viewing image through a lens placed over image
- Lens parameters: center ($x_c$, $y_c$ ), lens radius $r_{max}$ and refraction index $\rho$
- Sample values $\rho$ = 1.8 and $r_{max}$ = half image width

$$T_x^{-1}: \quad x = x' - \begin{cases} z \cdot \tan(\beta_x) & \text{for } r \le r_{\max} \\ 0 & \text{for } r > r_{\max}, \end{cases}$$

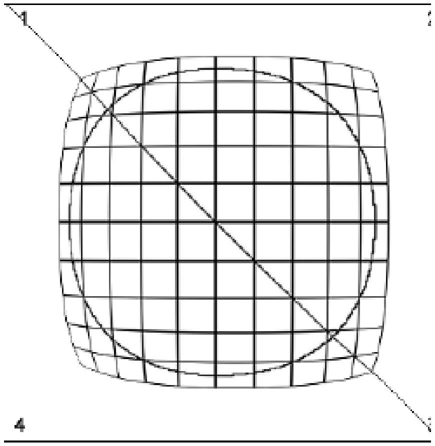
$$T_y^{-1}: \quad y = y' - \begin{cases} z \cdot \tan(\beta_y) & \text{for } r \le r_{\max} \\ 0 & \text{for } r > r_{\max}, \end{cases}$$

$$d_x = x' - x_c, \qquad r = \sqrt{d_x^2 + d_y^2}\,,$$

$$d_y = y' - y_c, \qquad z = \sqrt{r_{\max}^2 - r^2}\,,$$

$$\beta_x = \left(1 - \tfrac{1}{\rho}\right) \cdot \sin^{-1}\left(\frac{d_x}{\sqrt{(d_x^2 + z^2)}}\right),$$

$$\beta_y = \left(1 - \tfrac{1}{\rho}\right) \cdot \sin^{-1}\left(\frac{d_y}{\sqrt{(d_y^2 + z^2)}}\right).$$




(c)

(f)

# Image Warping