CSCE 448/748 – Computational Photography

Point Processing and Image Warping

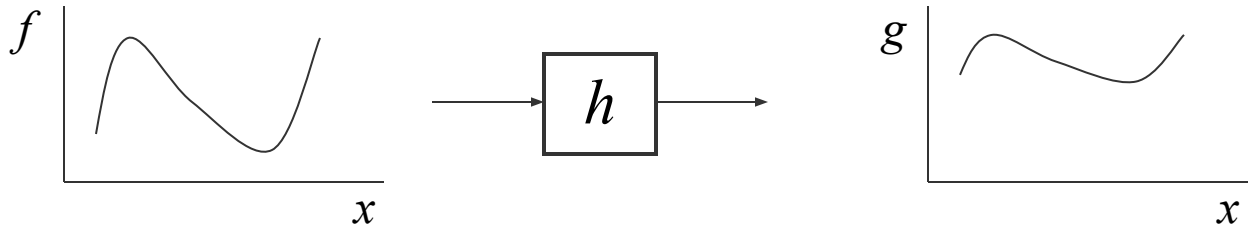
Nima Kalantari

Many slides from Alexei A. Efros, James Hayes

Image Processing

- **Point processing: change *range* of image**

$$g(x) = h(f(x))$$



- **Image warping: change domain of image**

$$g(x) = f(h(x))$$

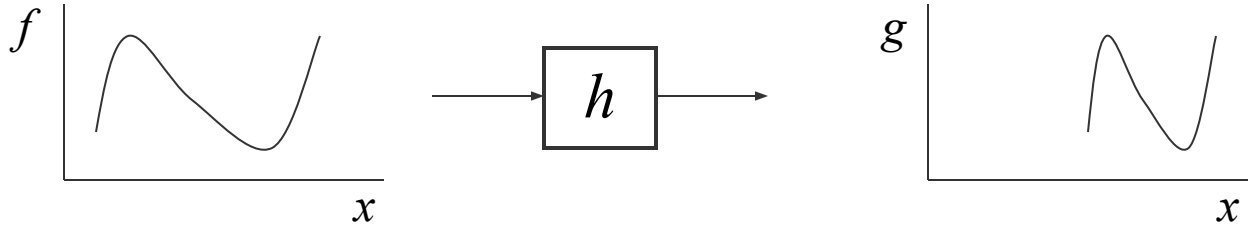
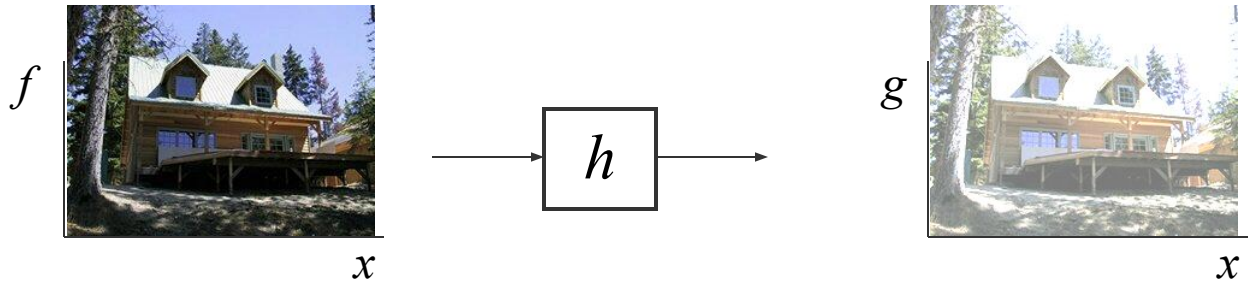


Image Processing

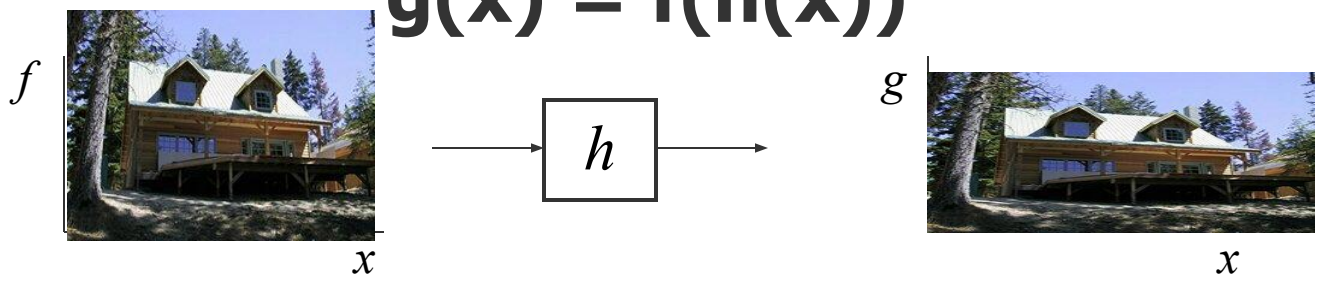
- **Point processing: change *range* of image**

$$g(x) = h(f(x))$$



- **Image warping: change domain of image**

$$g(x) = f(h(x))$$



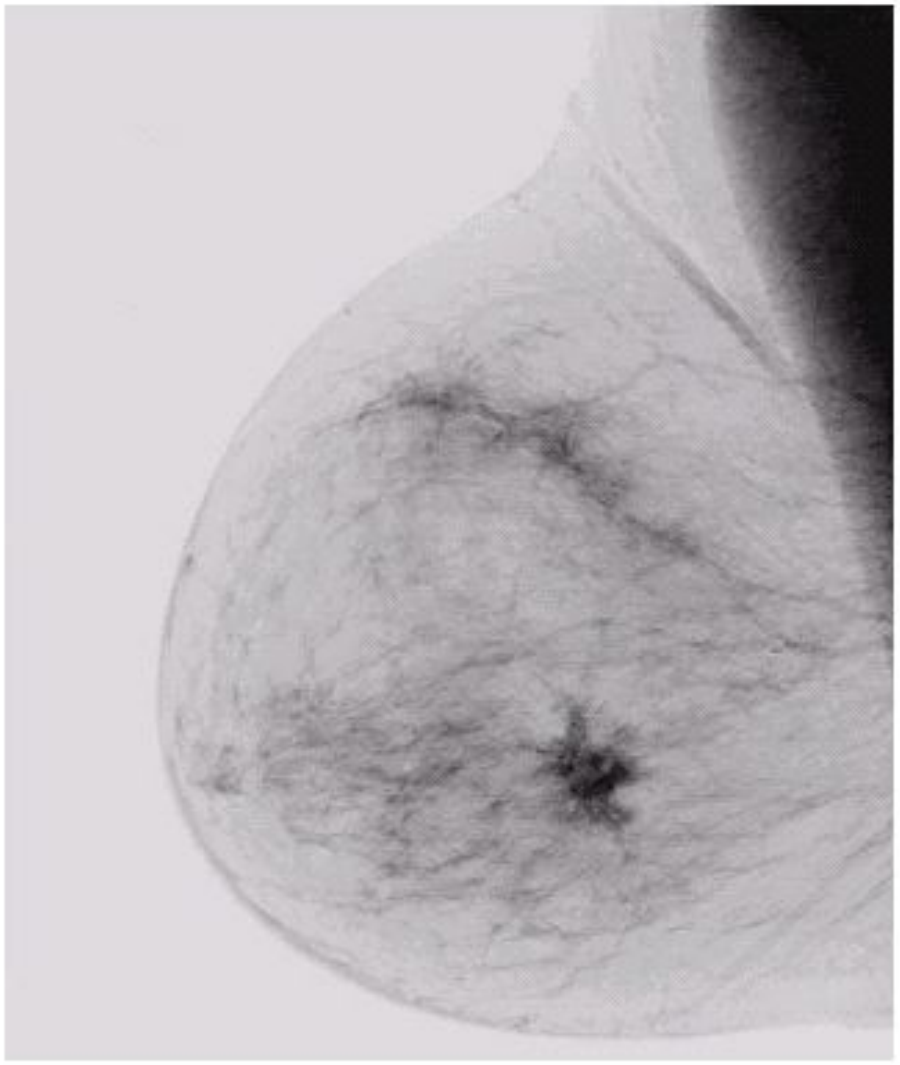
Point Processing

- The simplest kind of range transformations are independent of position x,y:

$$g = t(f)$$

- This is called point processing.
- What can they do?
- Important: every pixel for itself

Negative



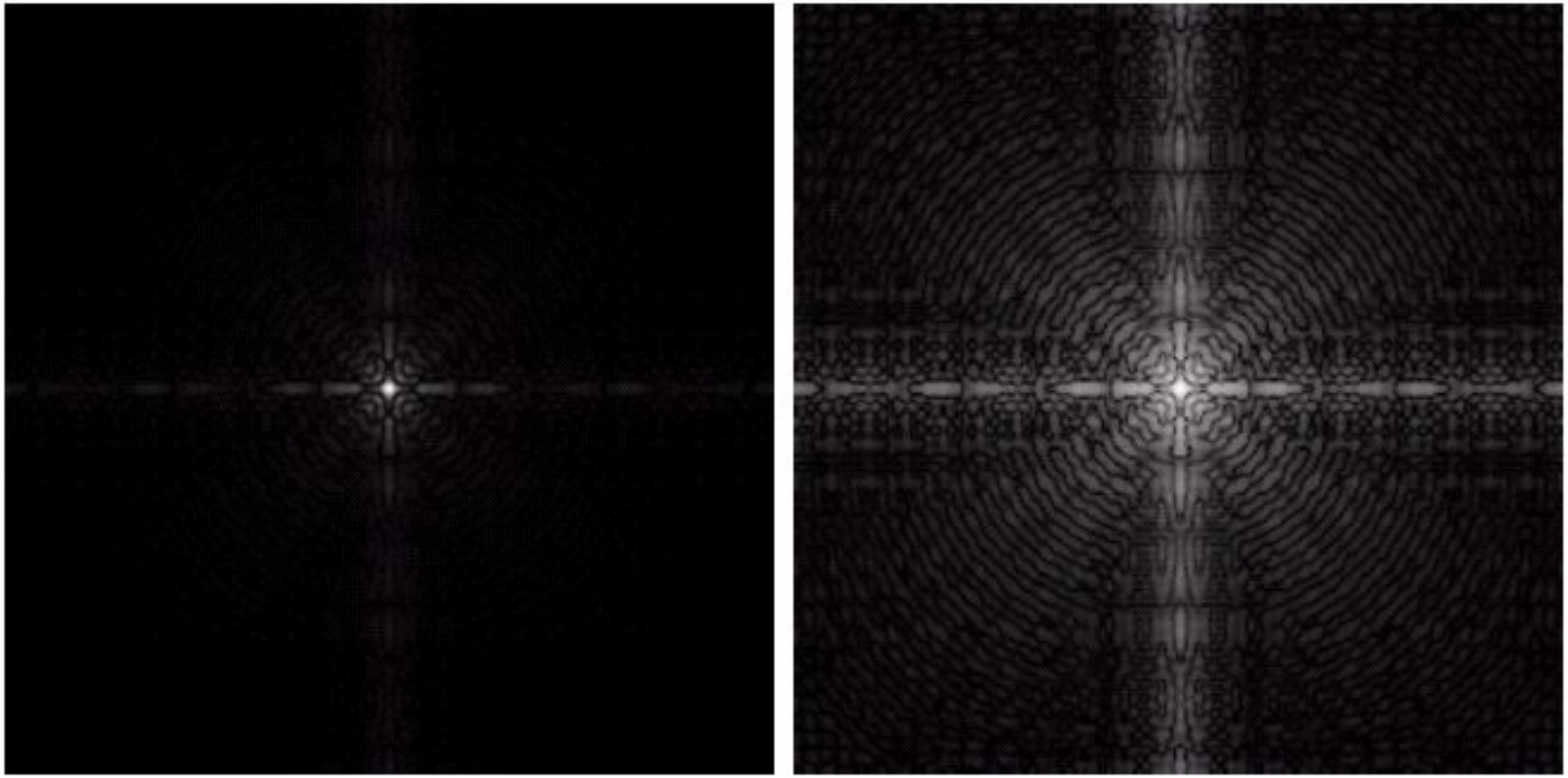
a b

FIGURE 3.4
(a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

Log

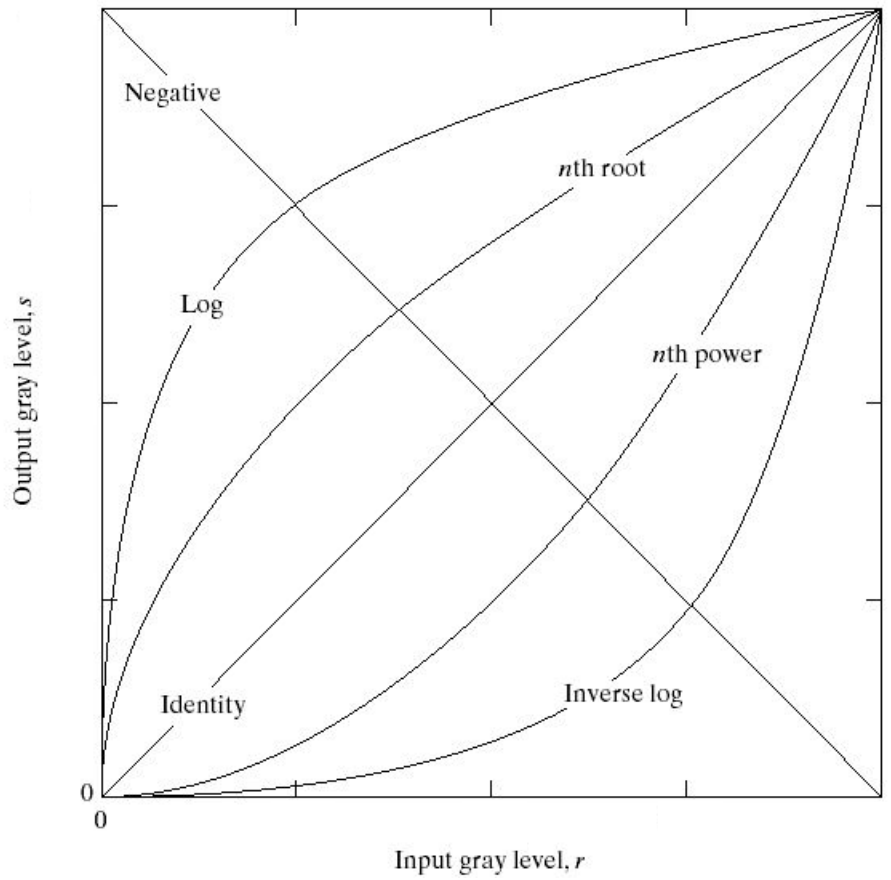
a b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.



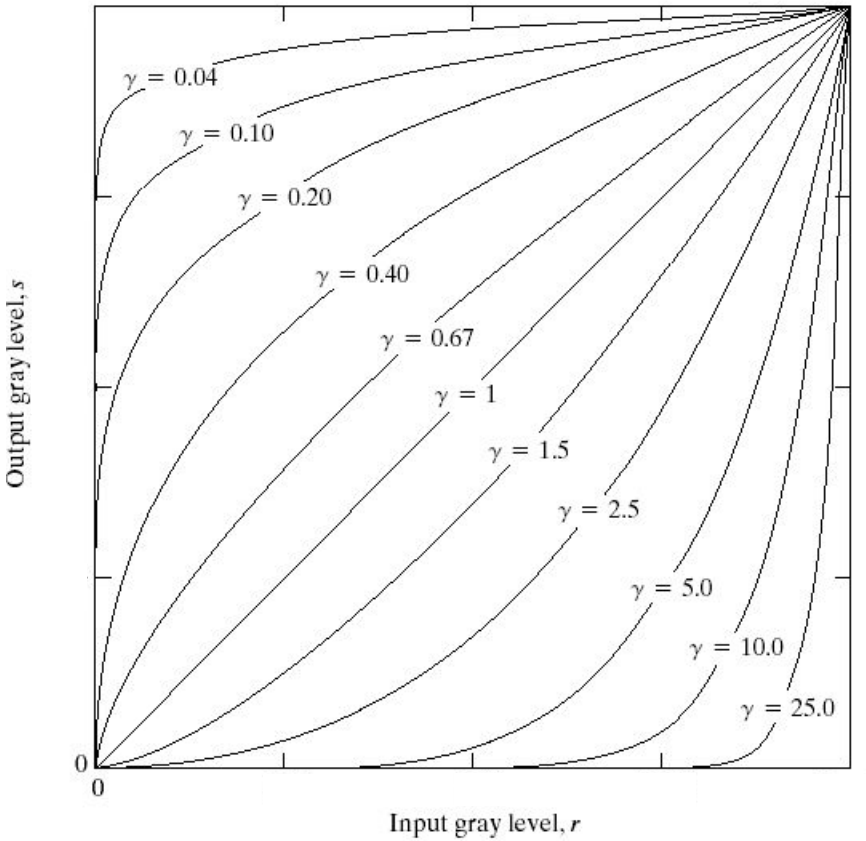
Power law functions

FIGURE 3.3 Some basic gray-level transformation functions used for image enhancement.



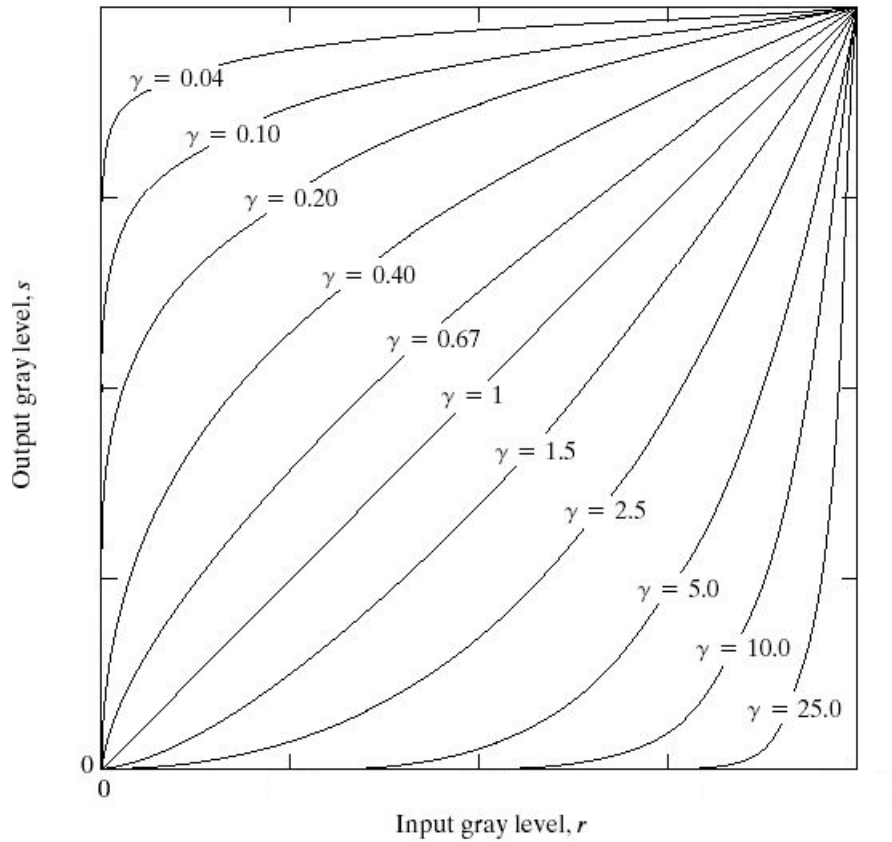
Gamma “correction”

- Power law intensity transformations
- Typically, gamma = 2.2 for a display device, and 1/2.2 for encoding.



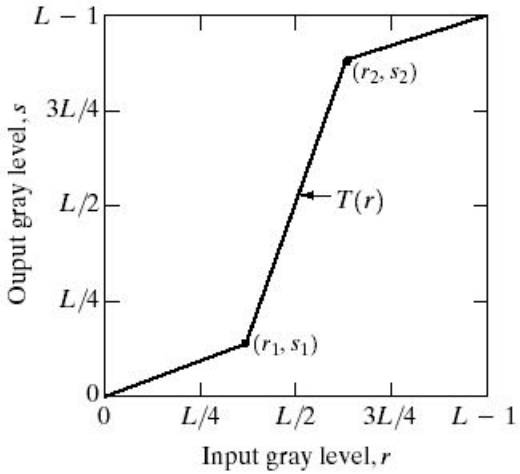
$$s = r^{\gamma}$$

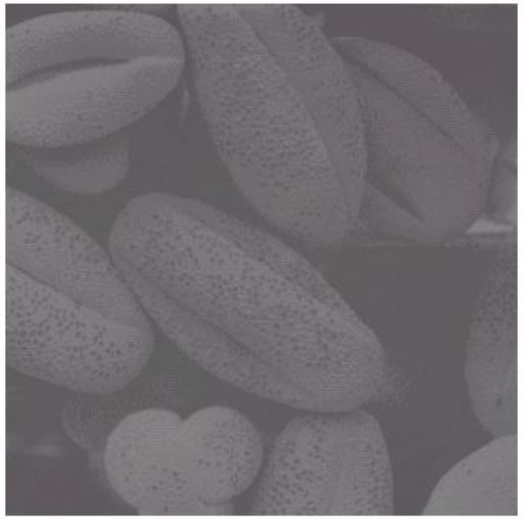
Gamma “correction”

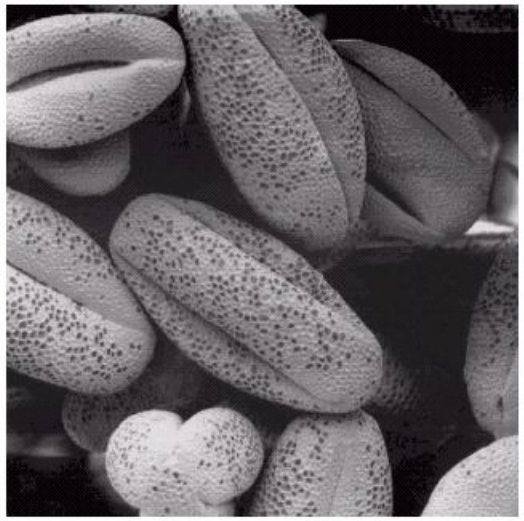


$$s = r^{\gamma}$$

Contrast Stretching



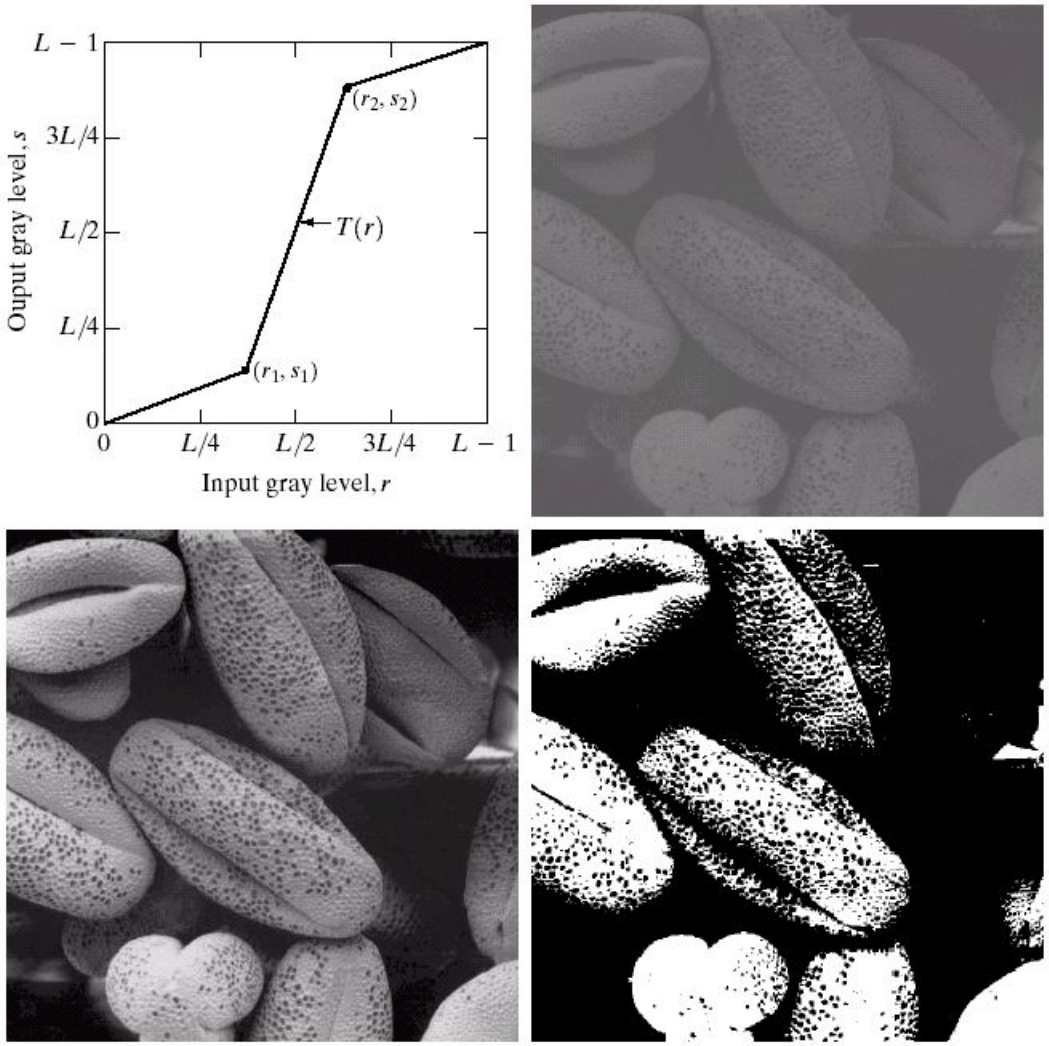




a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

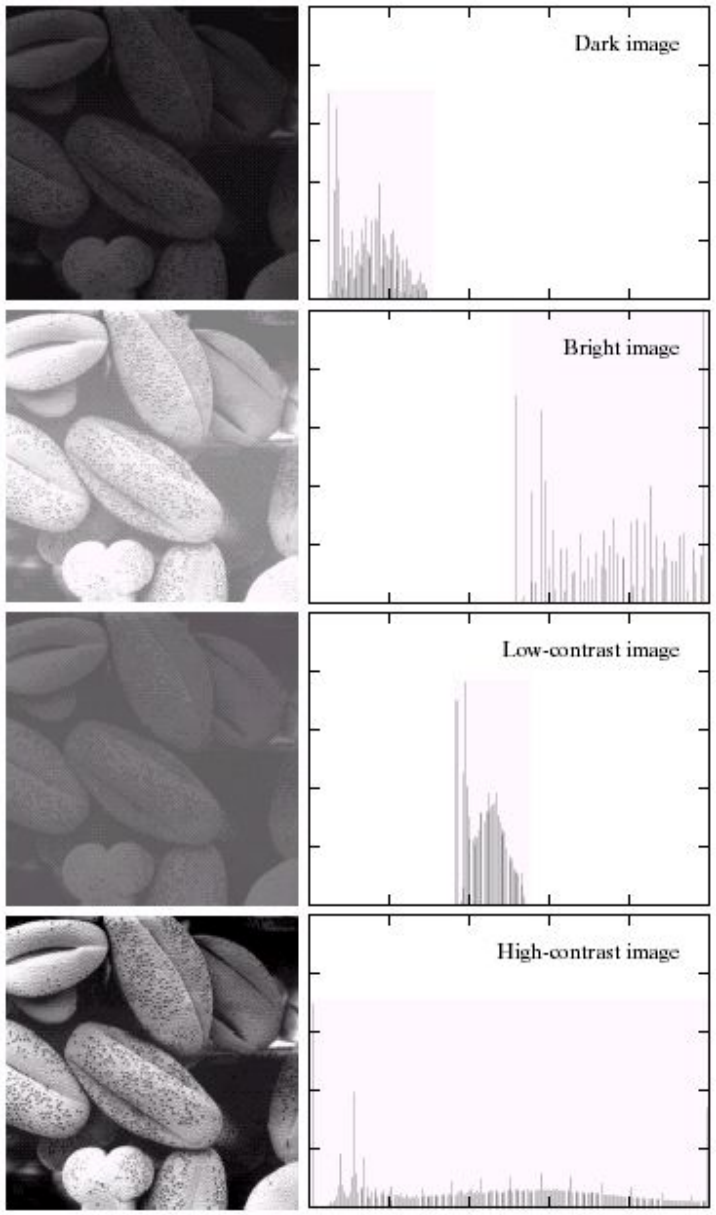
Contrast Stretching



a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Image Histograms

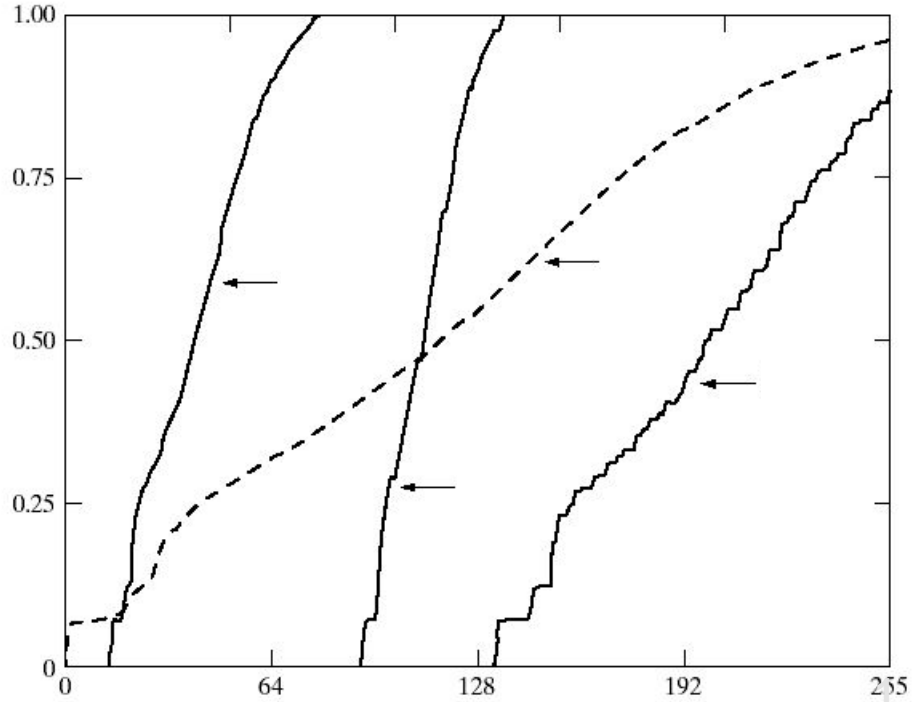


1

2

3

4



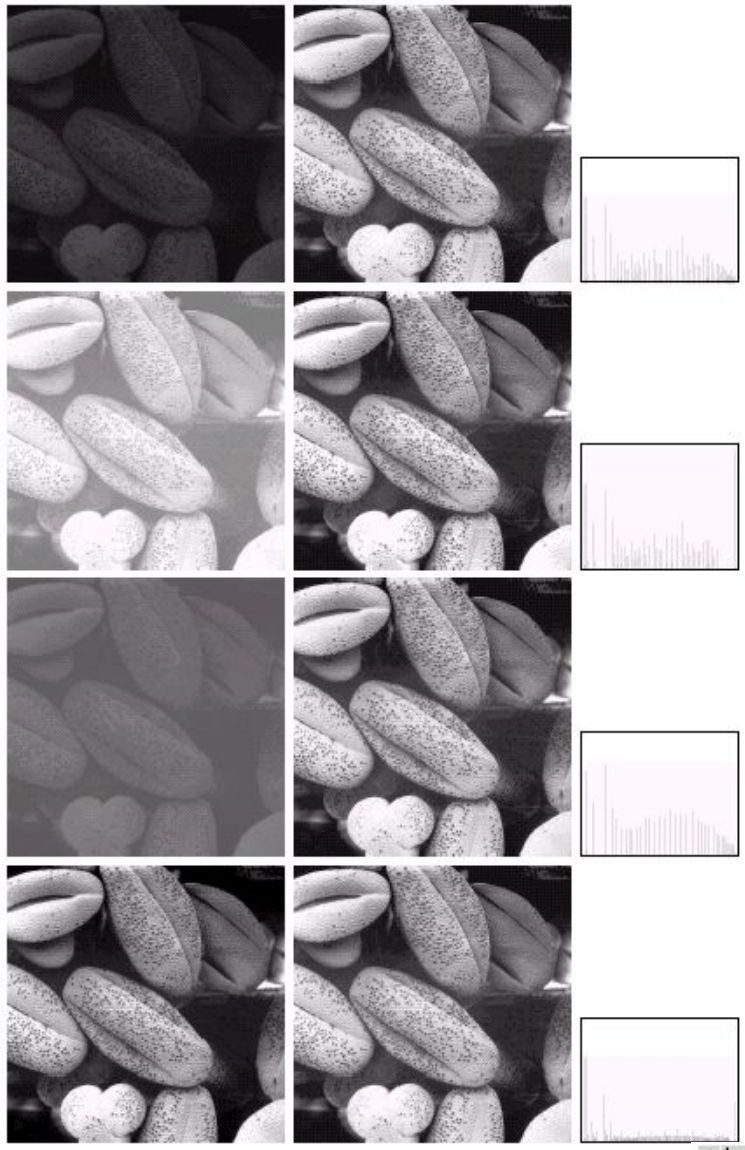
Cumulative Histograms

$s = T(r)$

a b

FIGURE 3.15 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Histogram Equalization



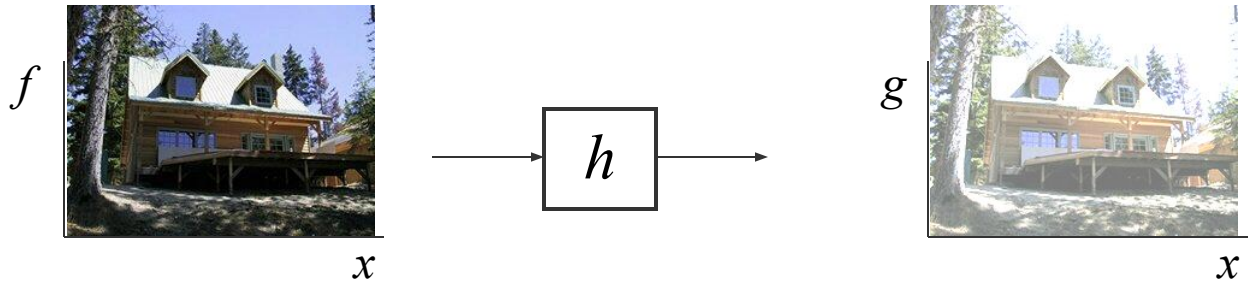
a b c

FIGURE 3.17 (a) Images from Fig. 3.15. (b) Results of histogram equalization. (c) Corresponding histograms.

Image Processing

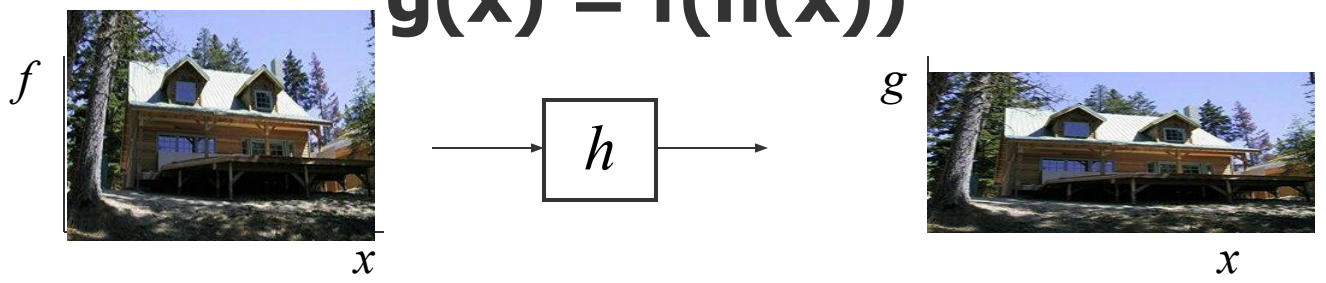
- **Point processing: change *range* of image**

$$g(x) = h(f(x))$$



- **Image warping: change domain of image**

$$g(x) = f(h(x))$$



Parametric (global) warping

- **Examples of parametric warps:**

translation

rotation

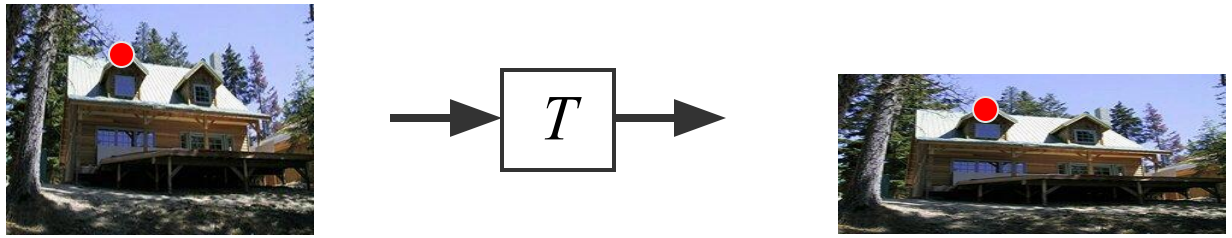
aspect

affine

perspective

cylindrical

Parametric (global) warping



$\mathbf{p} = (x,y)$ $\mathbf{p'} = (x',y')$

- **Transformation T is a coordinate-changing machine:**

$$\mathbf{p}' = T(\mathbf{p})$$

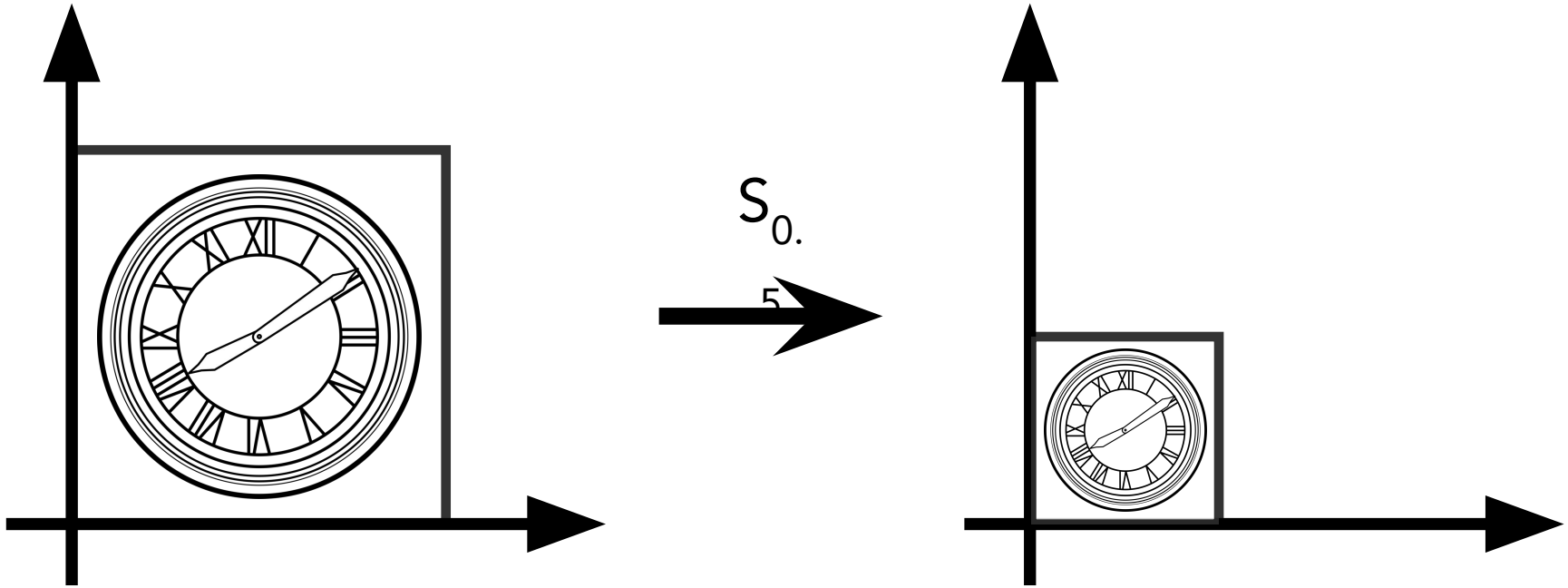
- **What does it mean that *T* is global and parametric?**
  - **Is the same for any point p**
  - **can be described by just a few numbers (parameters)**
- **Let's represent *T* as a matrix:**

$$\mathbf{p}' = \mathbf{M}\mathbf{p} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

Linear transforms

$$T(ax + by) = aT(x) + bT(y)$$

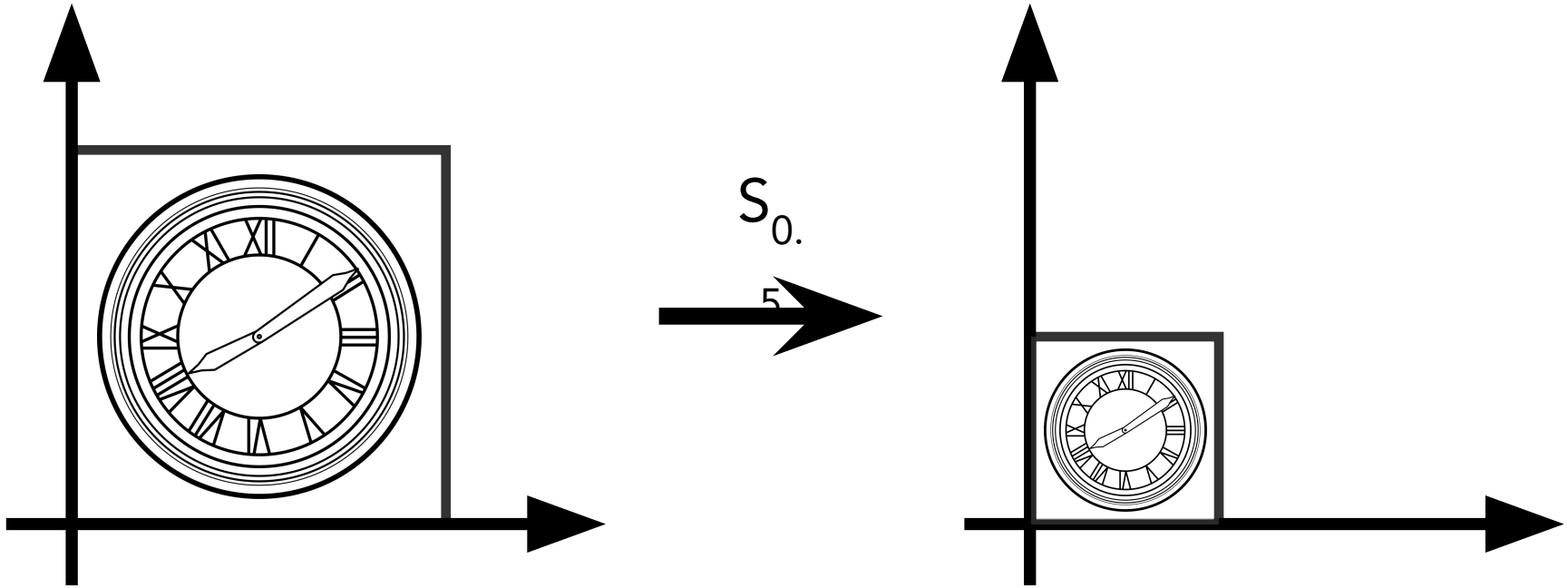
Scale Transform (Uniform)



$$x' = sx$$

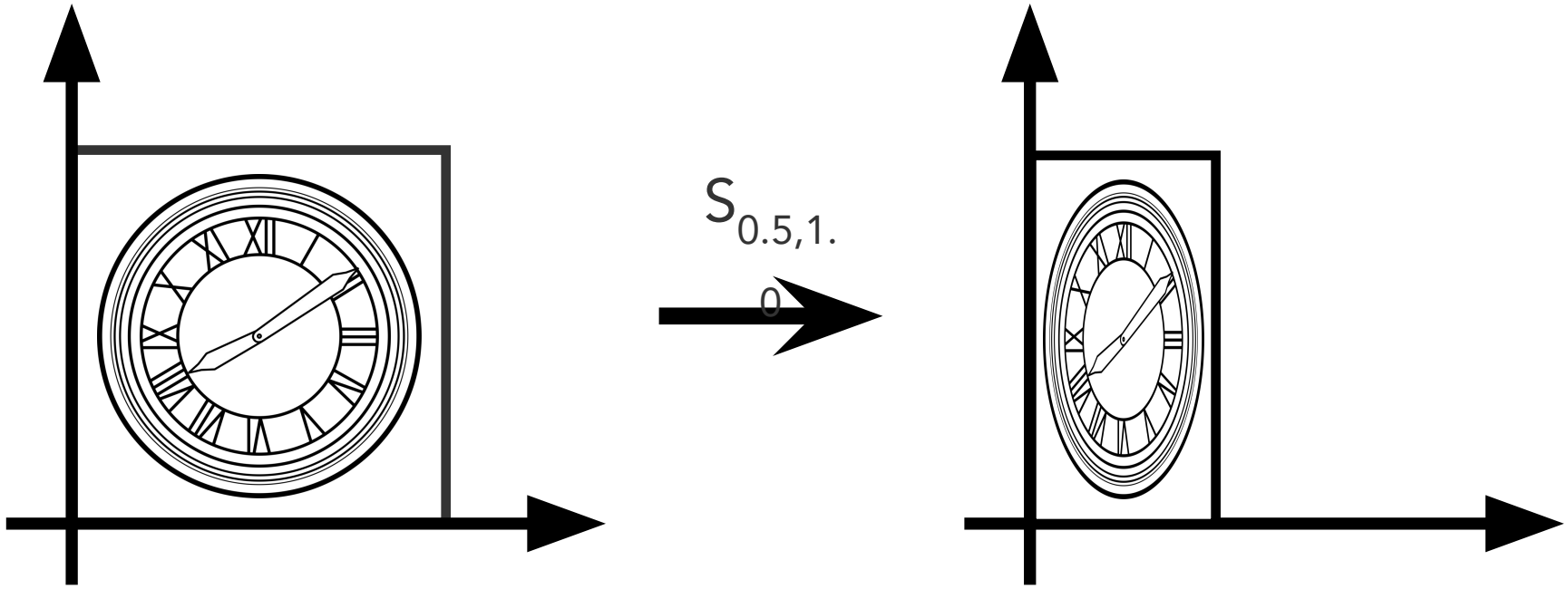
$$y' = sy$$

Scale Matrix (Uniform)



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale (Non-Uniform)



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

Scaling

- **Operation**

$$x' = s_x x$$
$$y' = s_y y$$

- **In matrix form**

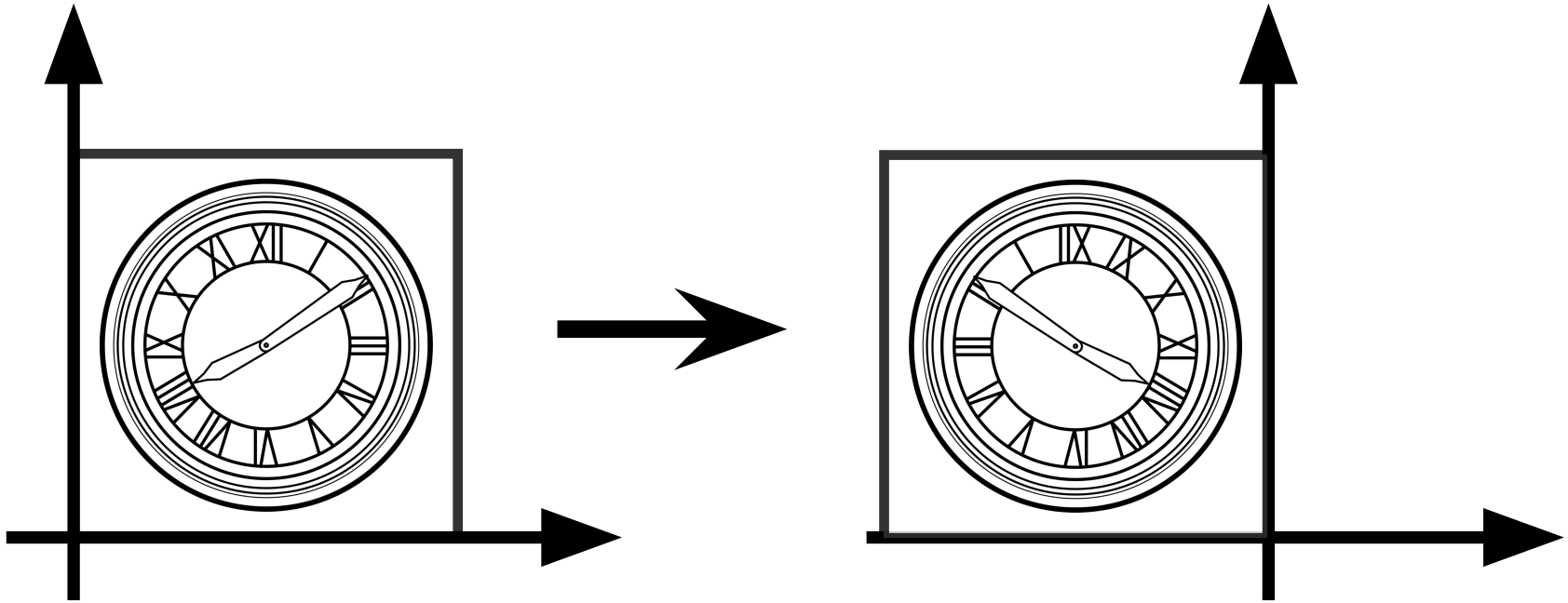
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- **What's the inverse?**

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

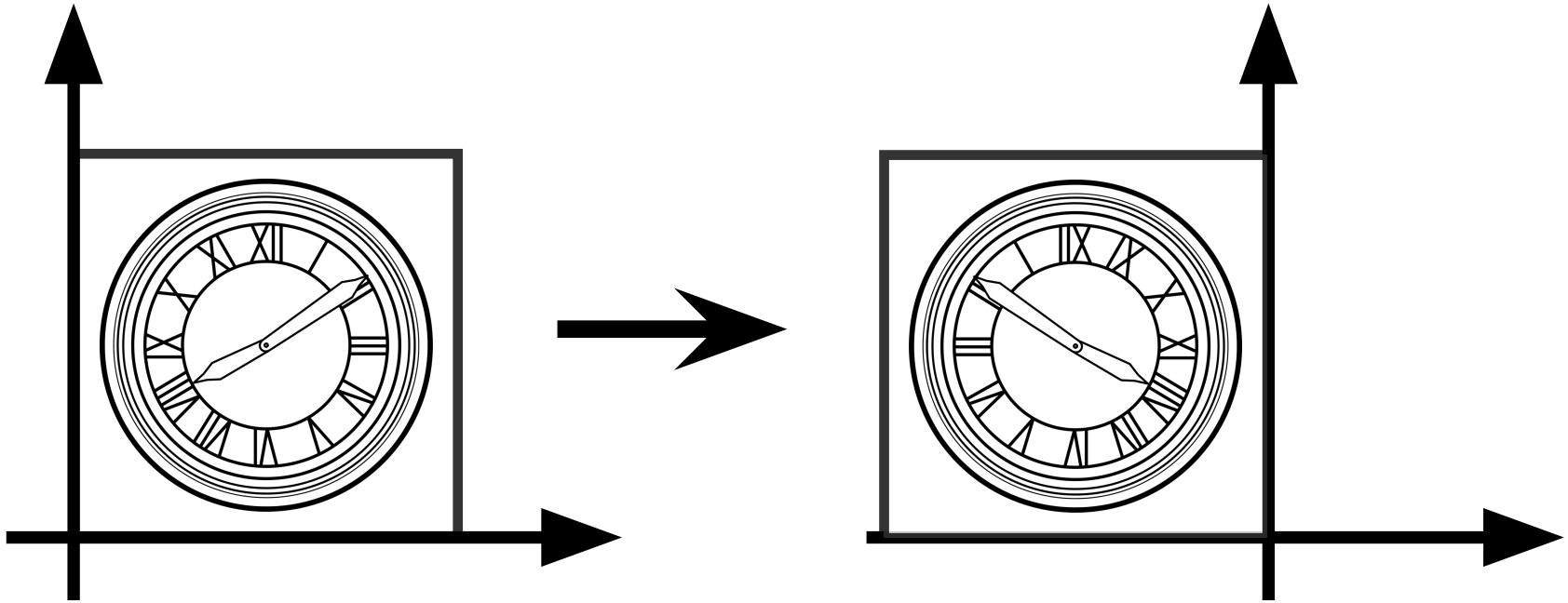
- **Is it linear?** **Yes!**

Reflection



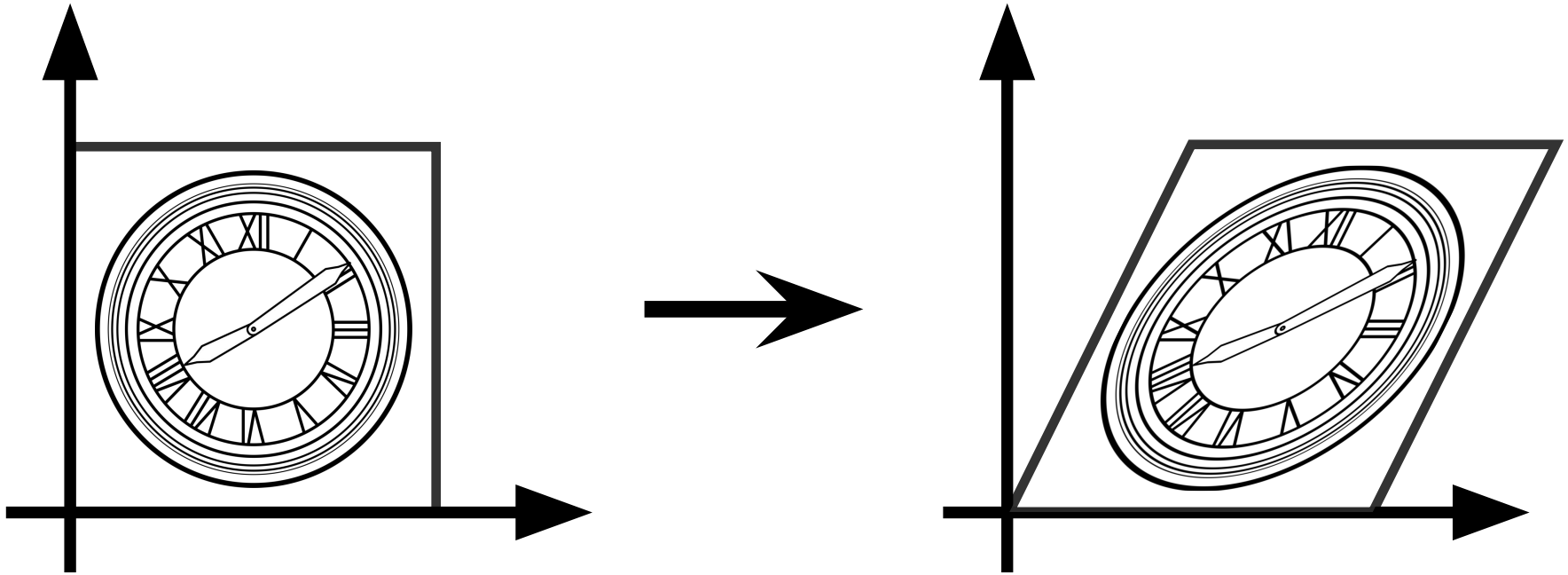
$$x' = -x$$
$$y' = y$$

Reflection Matrix



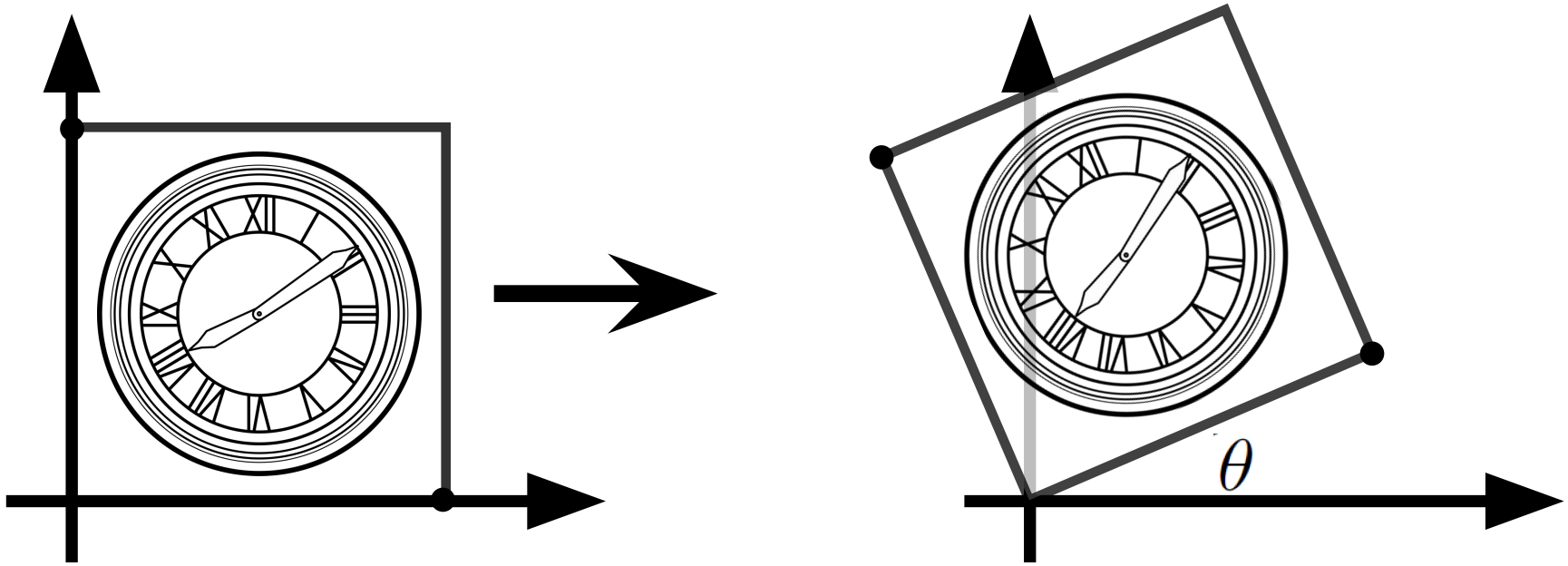
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear Matrix

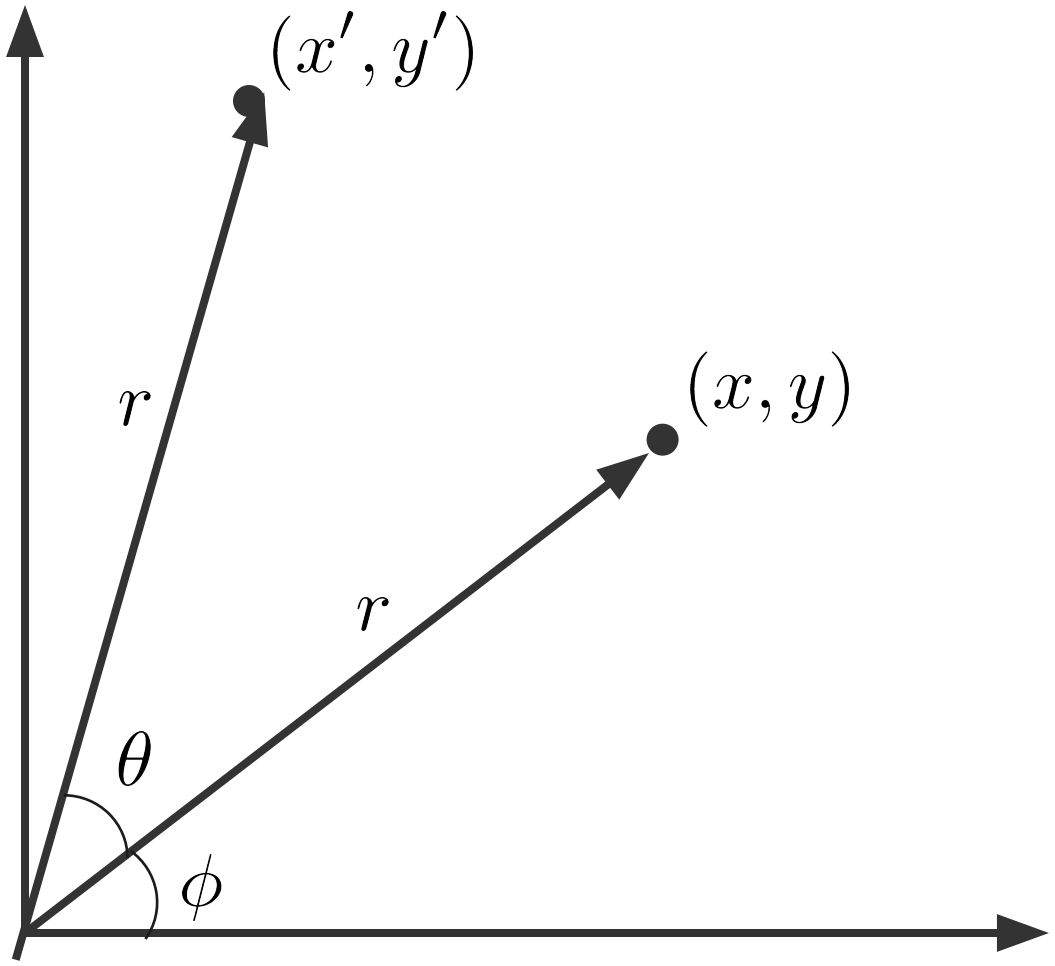


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

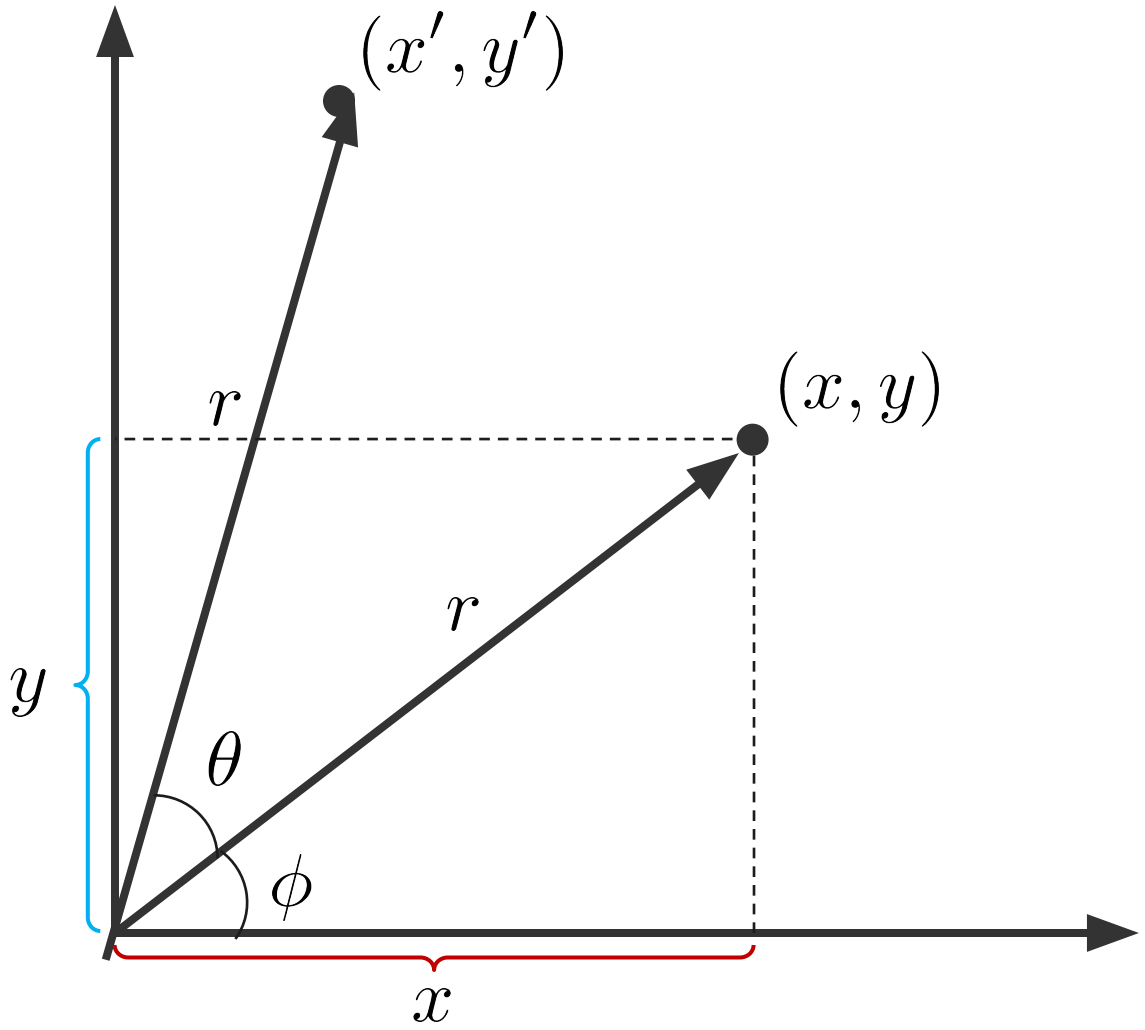
Rotation Matrix



2-D Rotation



2-D Rotation



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

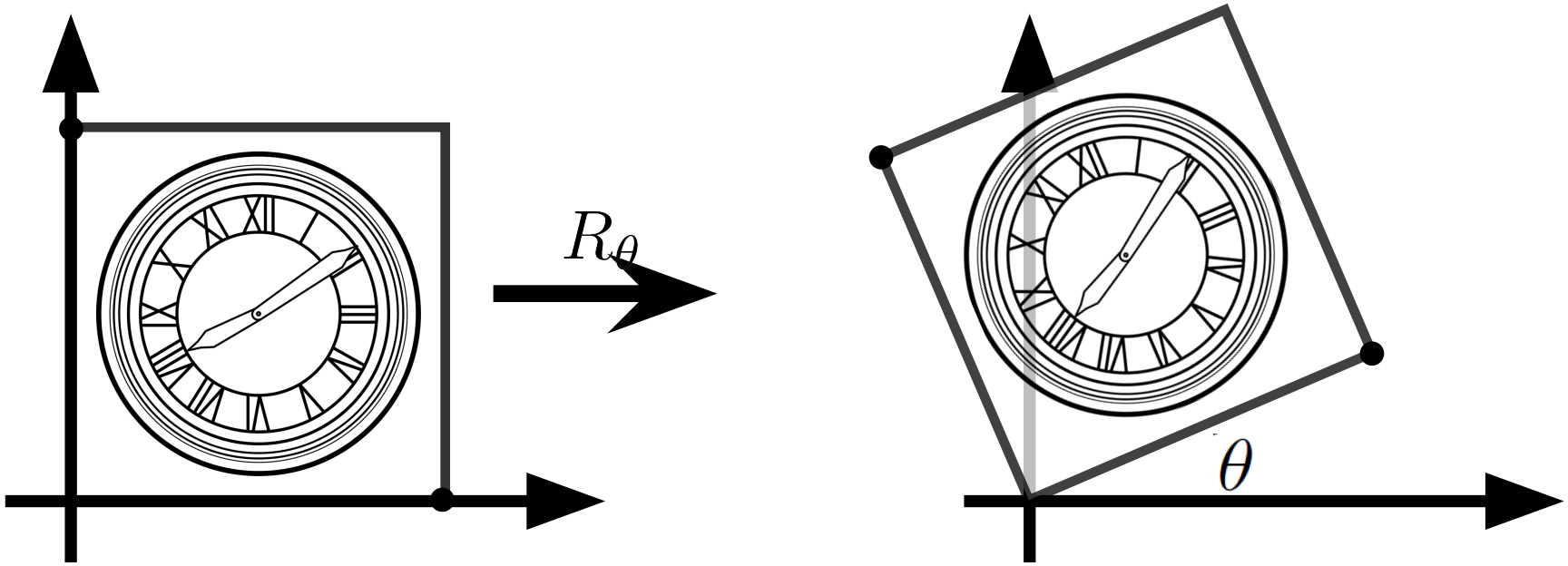
Trigonometric Identity

$$x' = \overbrace{r\cos(\phi)}^{x}\cos(\theta) - \overbrace{r\sin(\phi)}^{y}\sin(\theta)$$
$$y' = \underbrace{r\sin(\phi)}_{y}\cos(\theta) + \underbrace{r\cos(\phi)}_{x}\sin(\theta)$$

Substitute

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

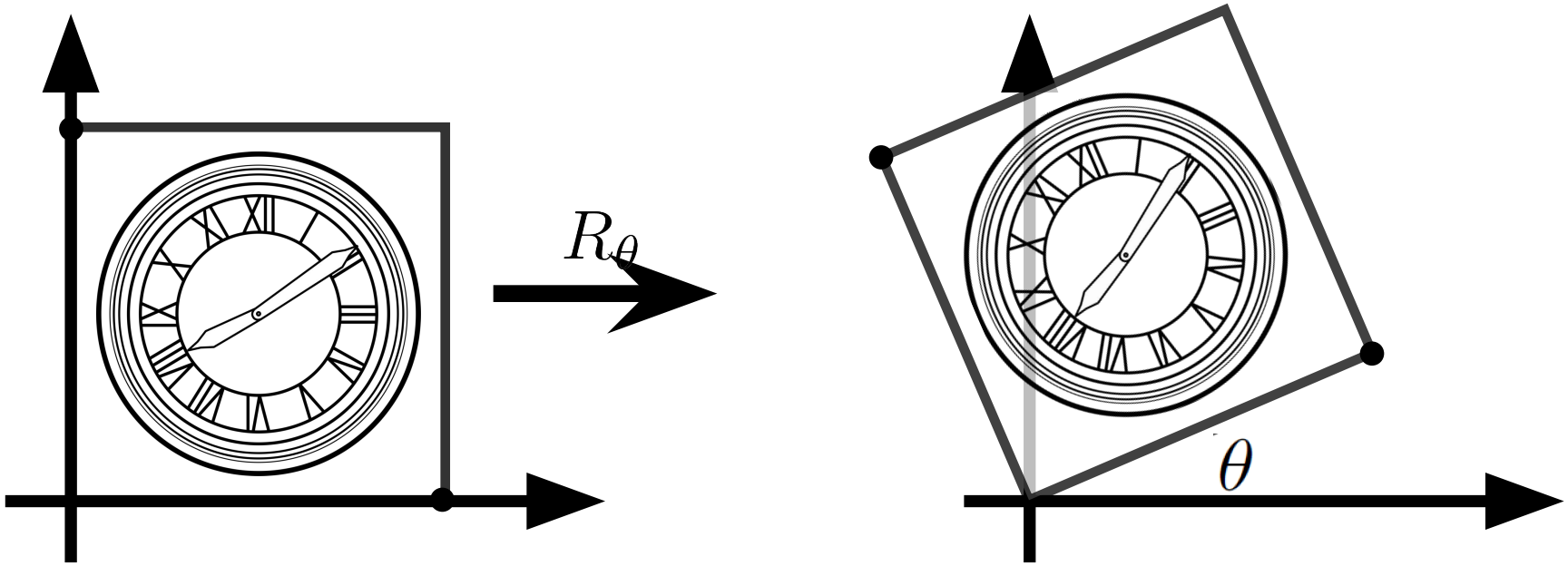
Rotation Matrix



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

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} . & . \\ . & . \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

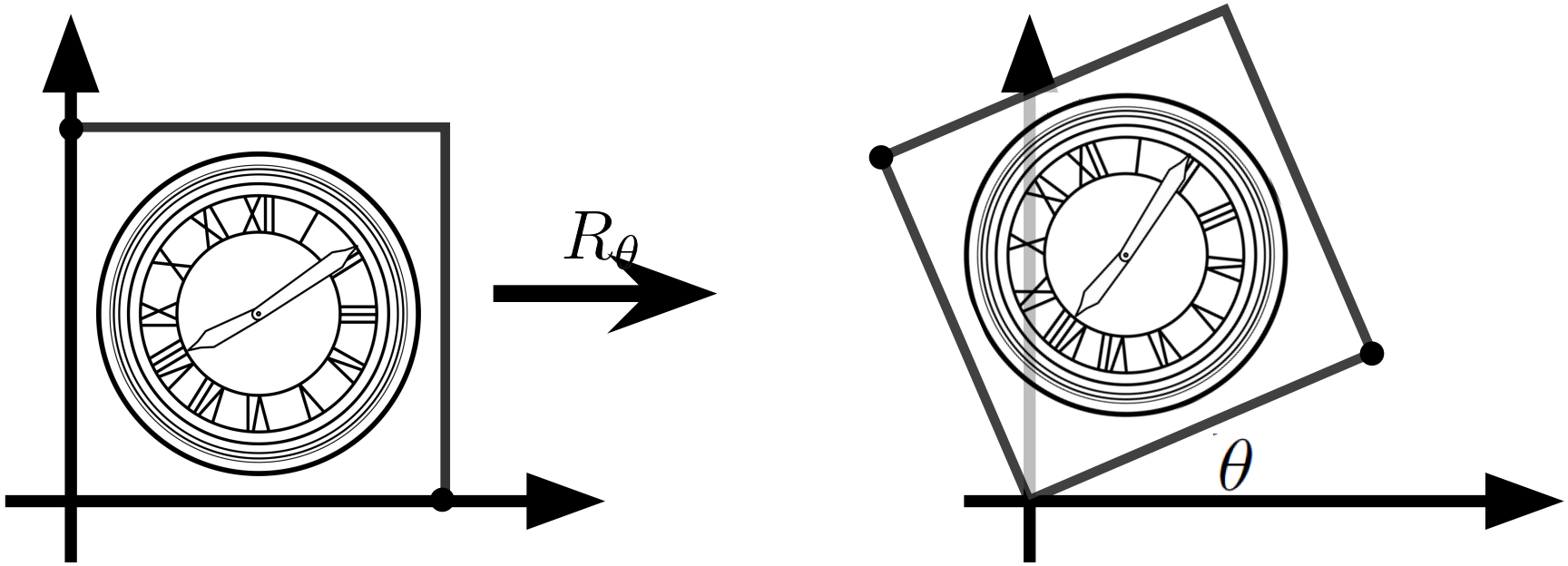
Rotation Matrix



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

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & . \\ . & . \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

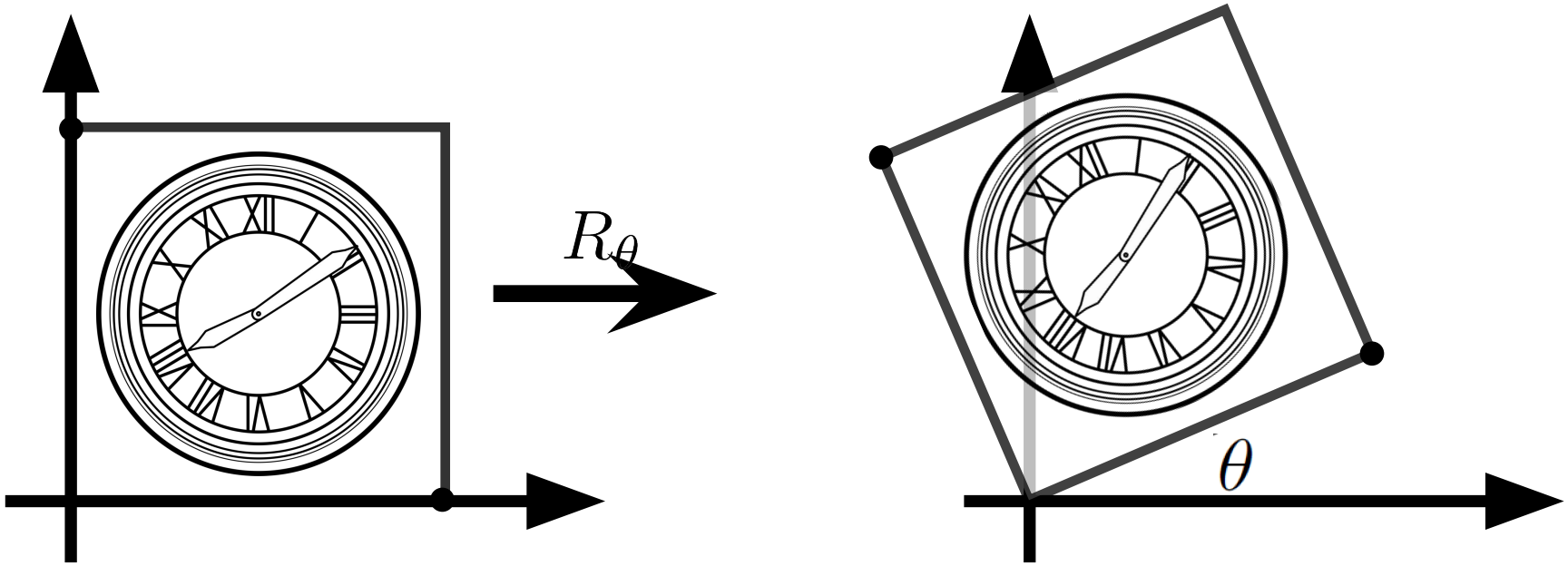
Rotation Matrix



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

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

Rotation Matrix



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

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \underbrace{\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}}_{R_\theta} \begin{pmatrix} x \\ y \end{pmatrix}$$

Rotation

- **Even though sin(θ) and cos(θ) are nonlinear functions of θ**
  - *x′ is a linear combination of x and y*
  - *y′ is a linear combination of x and y*
- **What's the inverse transformation?**
  - **Rotation by – θ**

$$R_\theta^{-1} = R_{-\theta} = R^T$$

Linear Transforms = Matrices

$$x' = a\,x + b\,y$$
$$y' = c\,x + d\,y$$

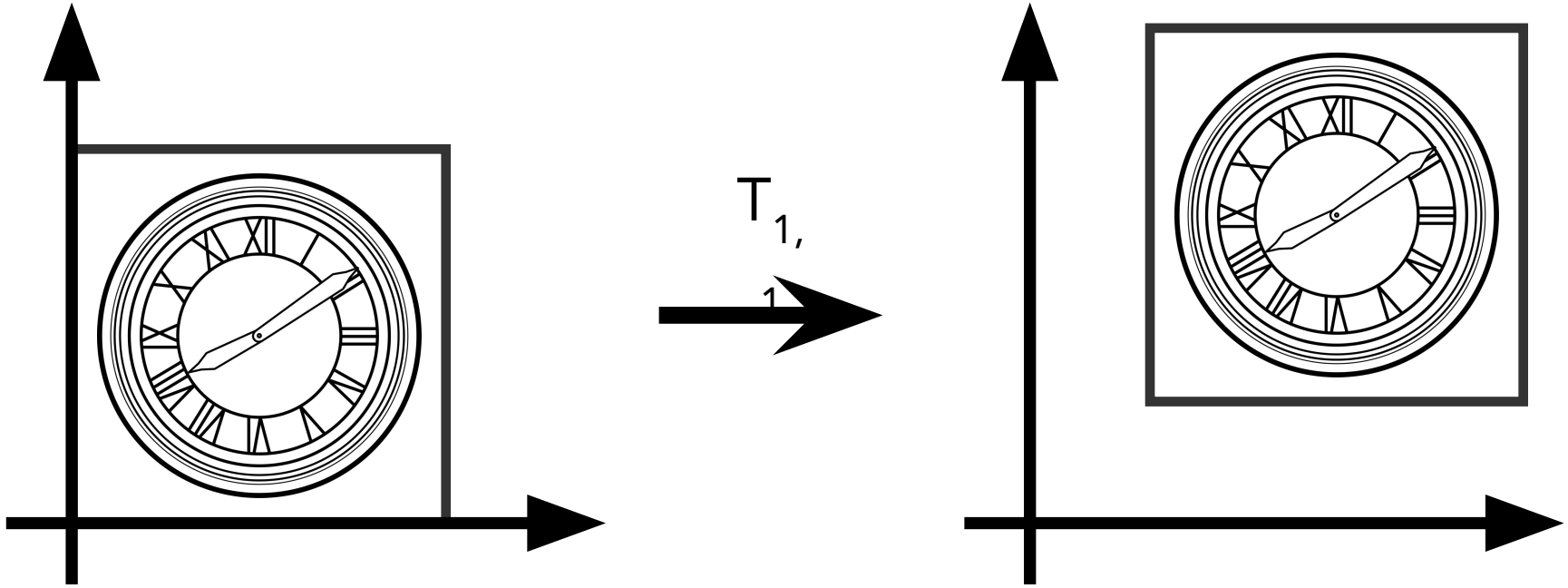
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{M}\mathbf{p}$$

All 2D Linear Transformations

- **Linear transformations are combinations of …**
  - **Scale,**
  - **Rotation,**
  - **Shear, and**
  - **Mirror**
- **Properties of linear transformations:**
  - **Origin maps to origin**
  - **Lines map to lines**
  - **Parallel lines remain parallel**
  - **Ratios of parallel line segments are preserved**
  - **Closed under composition**

Translation?



$$x' = x + t_x$$
$$y' = y + t_y$$

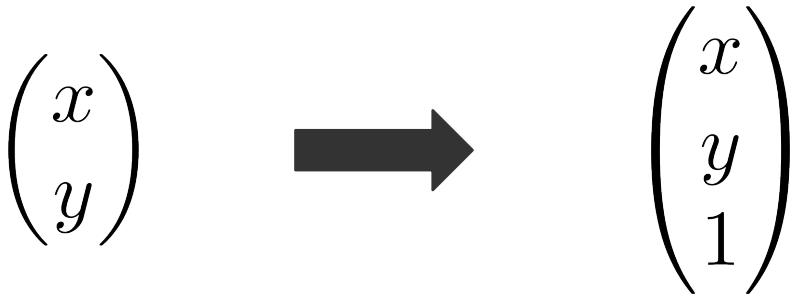
Translation

- E.g., move x by +5 units, leave y
- We need appropriate matrix. What is it?

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} ? & ? \\ ? & ? \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x+5 \\ y \end{pmatrix}$$

Homogeneous Coordinates

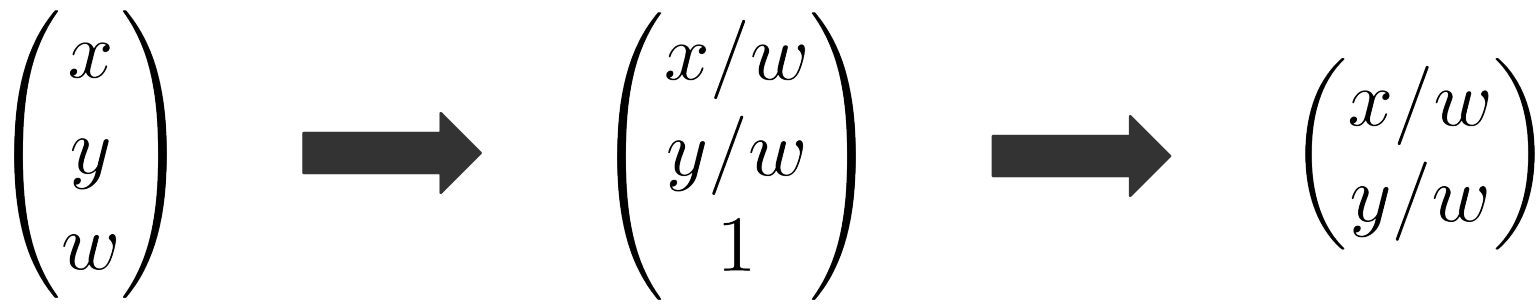
- Represent an n-dimensional Euclidean point using an n+1 dimensional point
- Set dimension n+1 (w) to 1
  - E.g., 2D Euclidean -> 3D Homogeneous



Euclidean Homogeneous

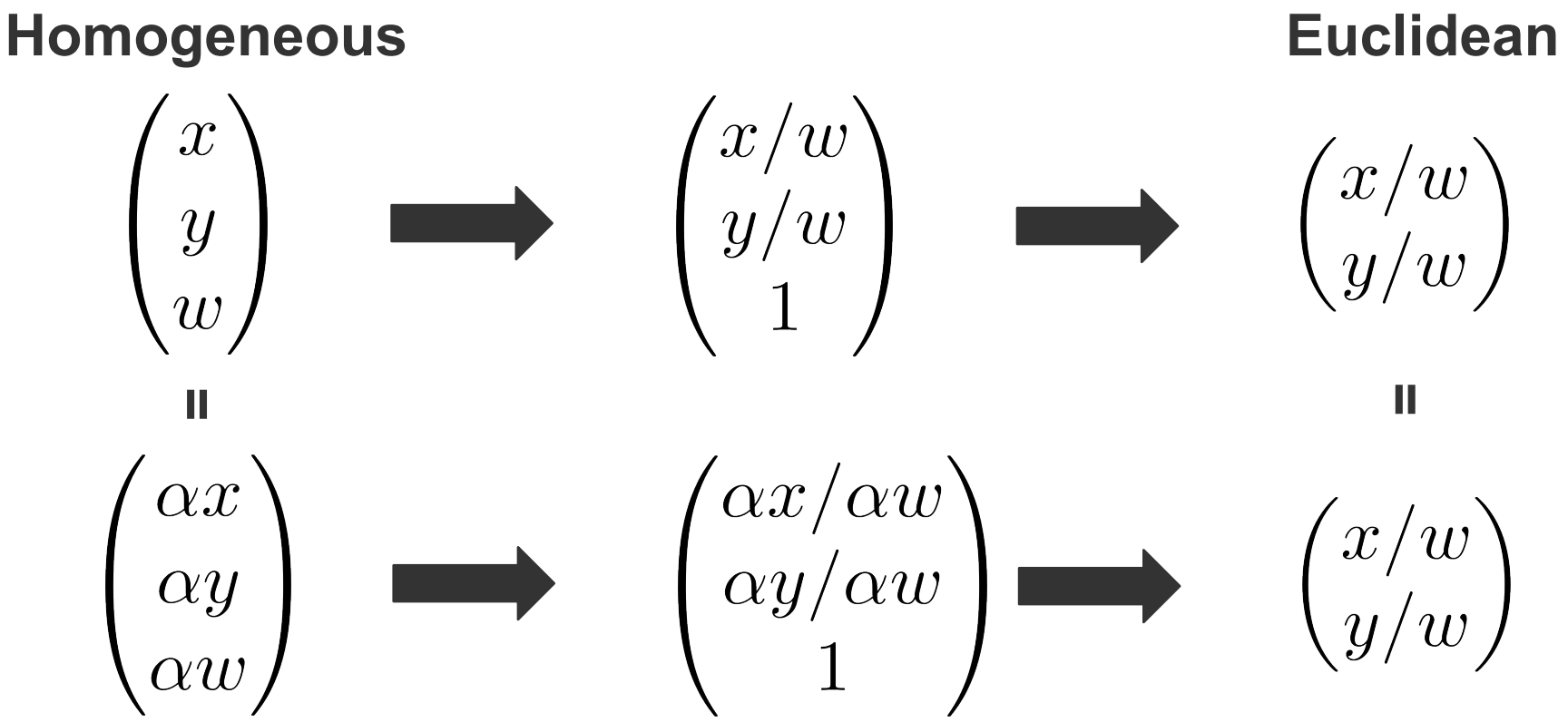
Homogeneous Coordinates

- To go from homogeneous coordinate to Euclidean
  - Divide by the last coordinate
  - Remove the normalized last coordinate

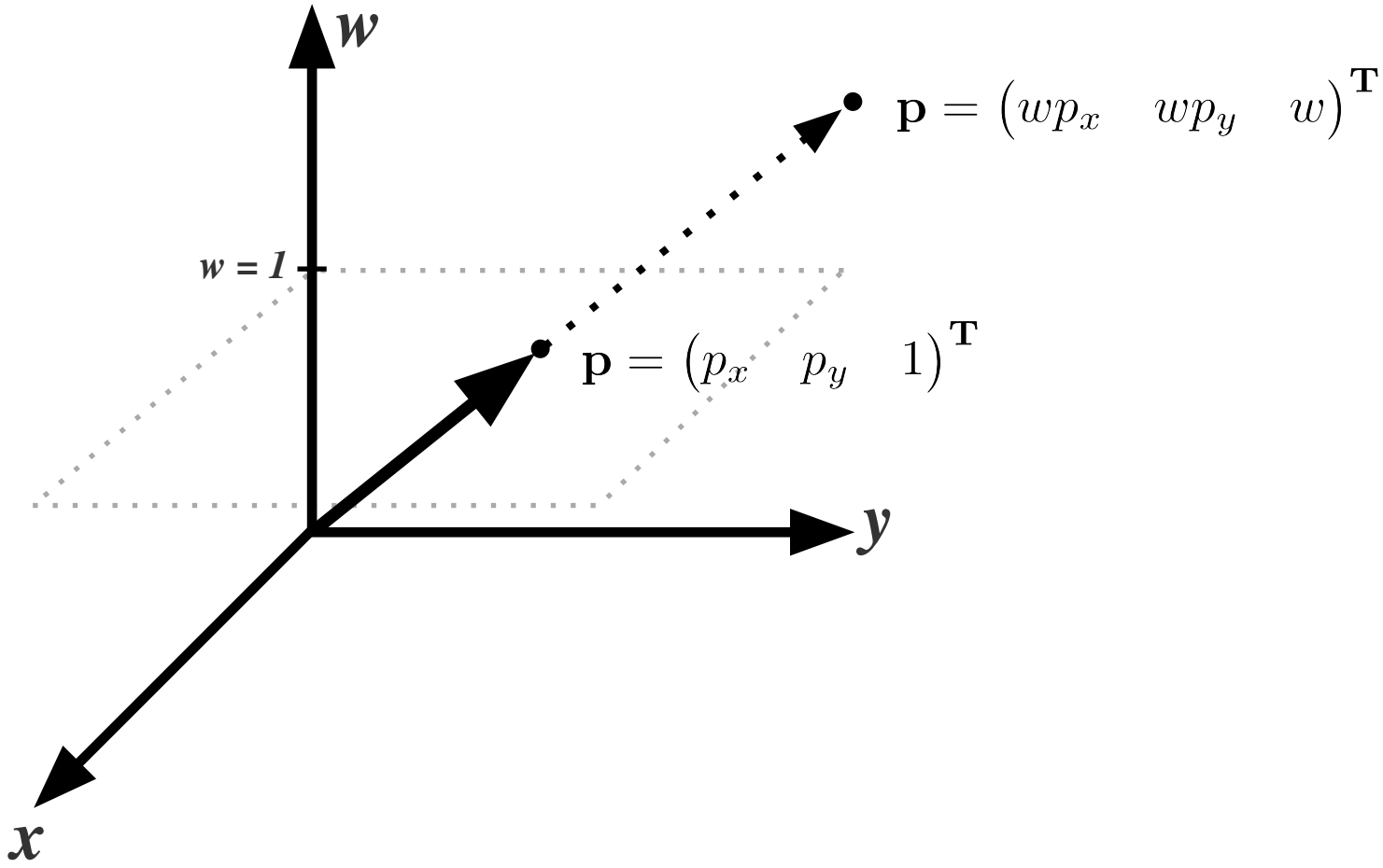


Homogeneous Coordinates

- **Multiplication by scalar has no effect**



Homogeneous coordinates: some intuition



Translation in Homogeneous Coordinates

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+5 \\ y \\ 1 \end{pmatrix}$$

Translation in Homogeneous Coordinates

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+5 \\ y \\ 1 \end{pmatrix}$$

General Translation Matrix

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

2D Transformations as 3x3 matrices

- **Scale**

$$\mathbf{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Rotation**

$$\mathbf{R}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **Translation**

$$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Matrix Composition

- **Combine transformations by matrix multiplication**

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{p'} = T(t_x,t_y) \quad R(\Theta) \quad S(s_x,s_y) \quad \mathbf{p}$$

1) Scale

2) Rotate

3) Translate

Affine Transformations

- **Affine transformations are combinations of …**
  - **Linear transformations, and**
  - **Translations**

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- **Properties of affine transformations:**
  - **Origin does not necessarily map to origin**
  - **Lines map to lines**
  - **Parallel lines remain parallel**
  - **Ratios of parallel line segments are preserved**
  - **Closed under composition**

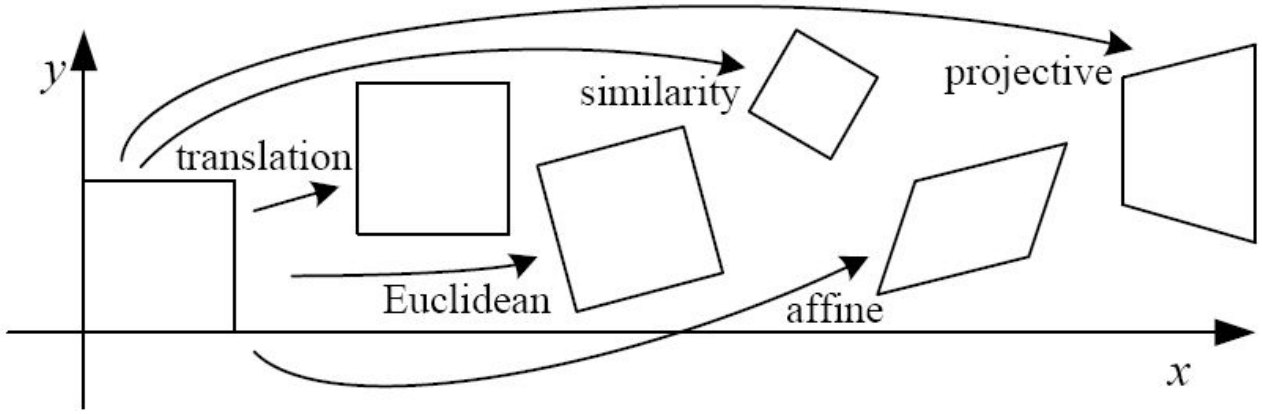
- **Will the last coordinate *w* always be 1?**

Projective Transformations

- **Projective transformations …**
  - **Affine transformations, and**
  - **Projective warps**
- **Properties of projective transformations:**
  - **Origin does not necessarily map to origin**
  - **Lines map to lines**
  - **Parallel lines do not necessarily remain parallel**
  - **Ratios are not preserved**
  - **Closed under composition**

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D image transformations



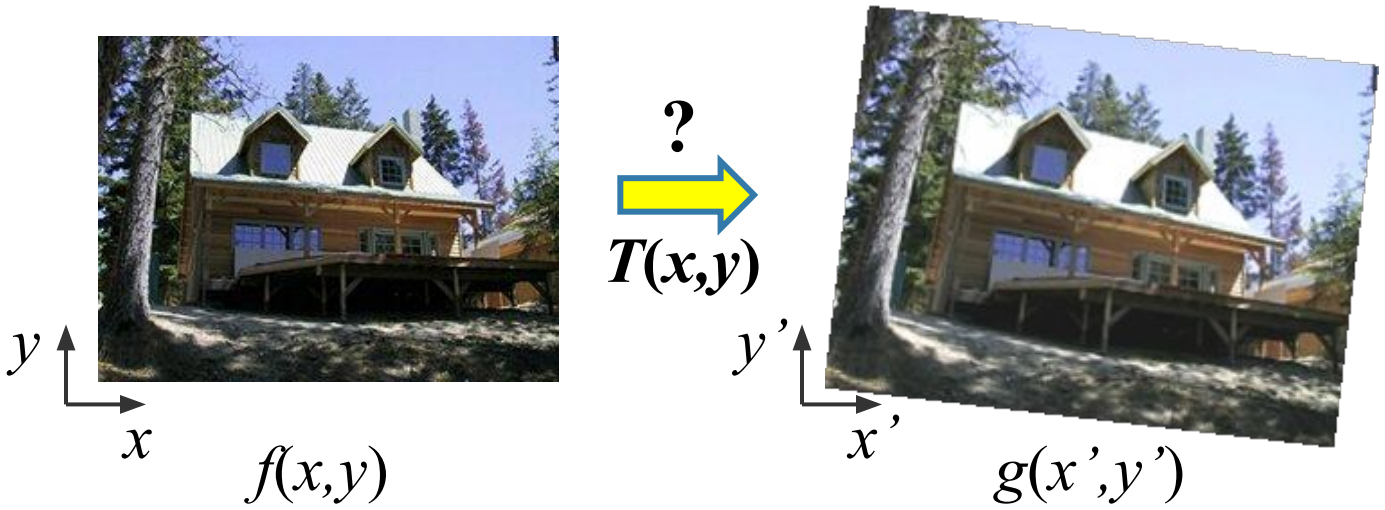
Name	# D.O.F.	Preserves:	Icon
translation			
rigid (Euclidean)			
similarity			
affine			
projective			

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Recovering Transformations

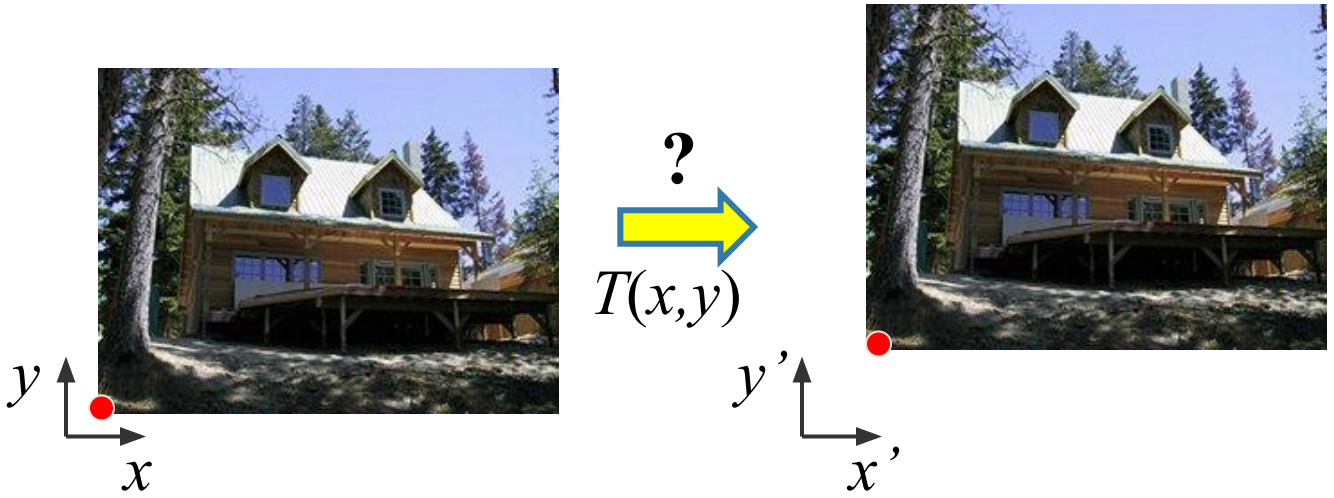
- What if we know *f* and *g* and want to recover the transform T?
  - e.g., better align images from Project 2
  - Willing to let user provide correspondences
    - How many do we need?



Translation: # correspondences?

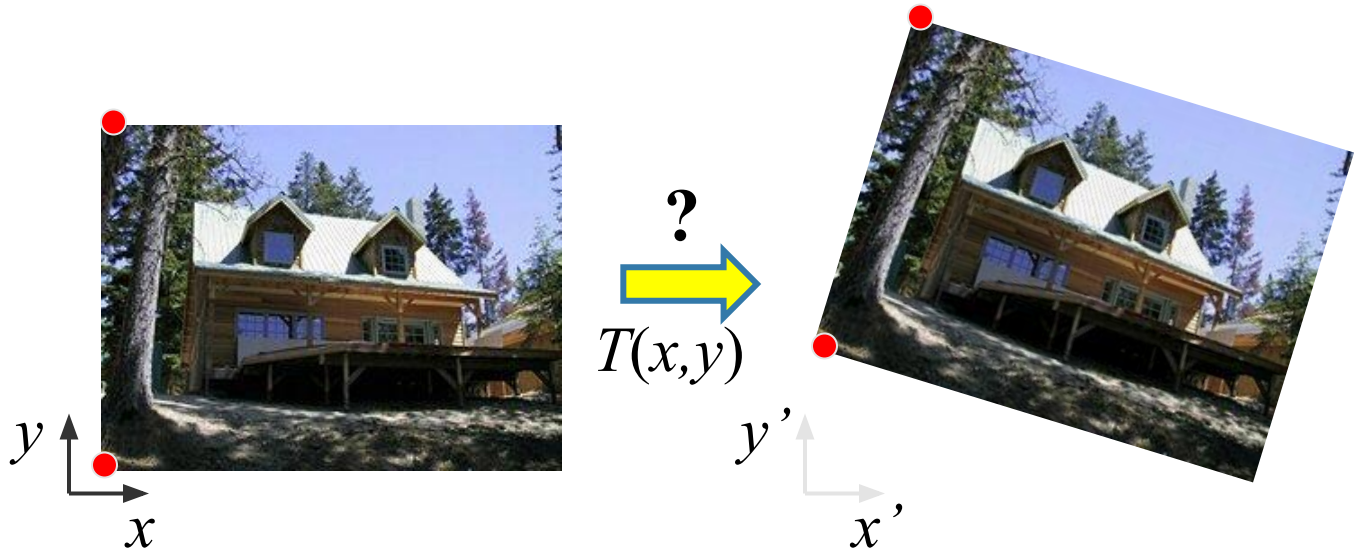
- How many correspondences needed for translation?
- How many Degrees of Freedom?
- What is the transformation matrix?

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$



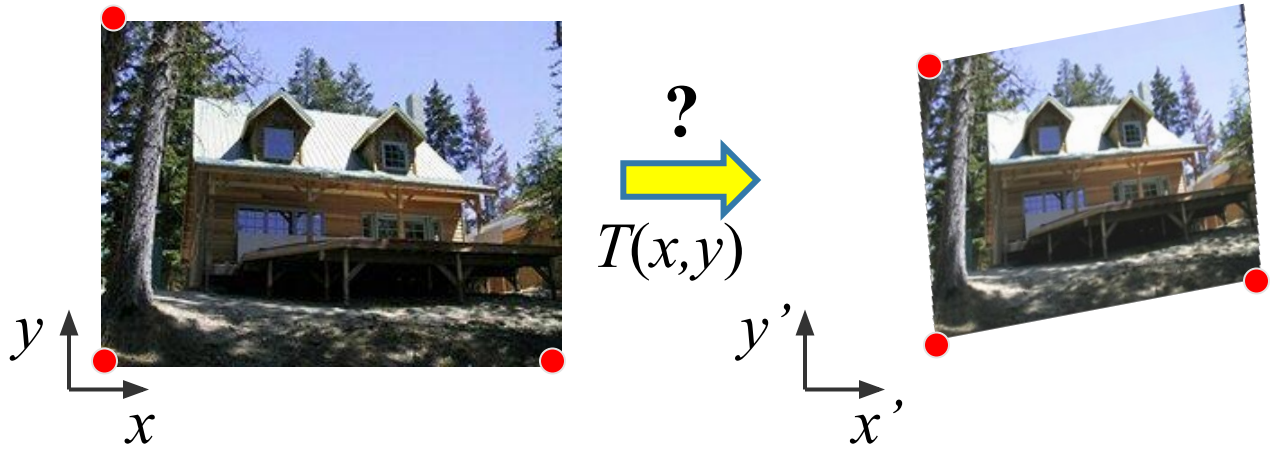
Euclidian: # correspondences?

- How many correspondences needed for translation+rotation?
- How many DOF?



Affine: # correspondences?

- How many correspondences needed for affine?
- How many DOF?



Projective: # correspondences?

- How many correspondences needed for projective?
- How many DOF?

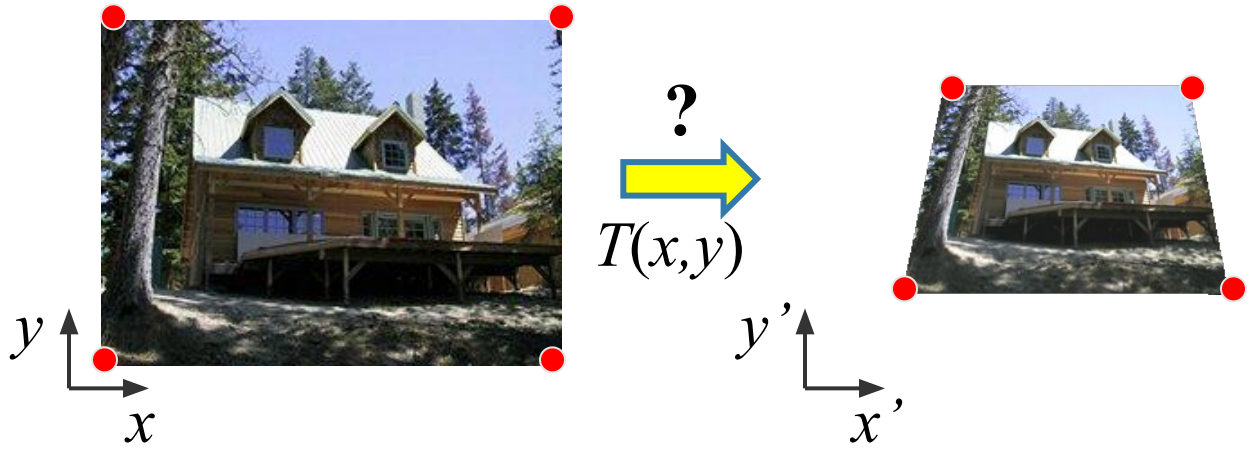
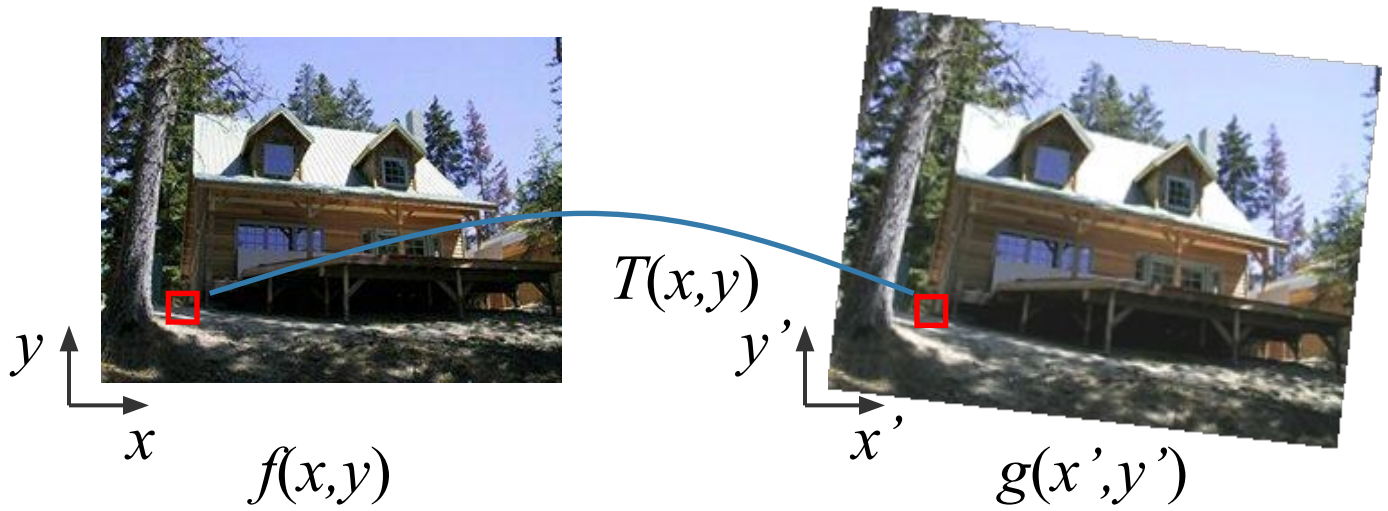


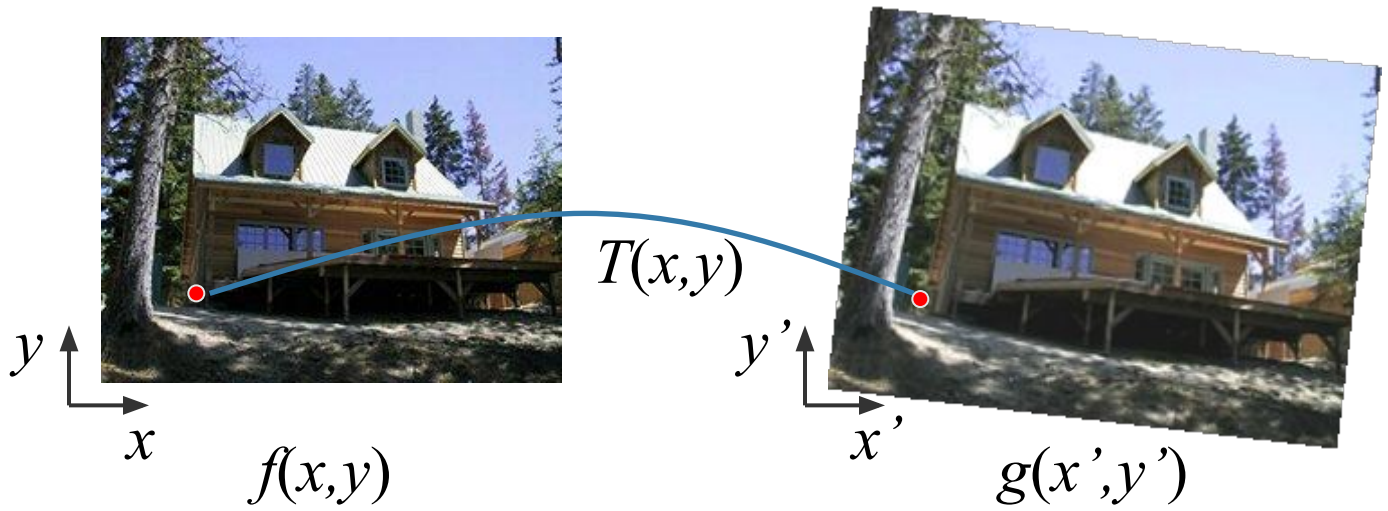
Image warping

- Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image?



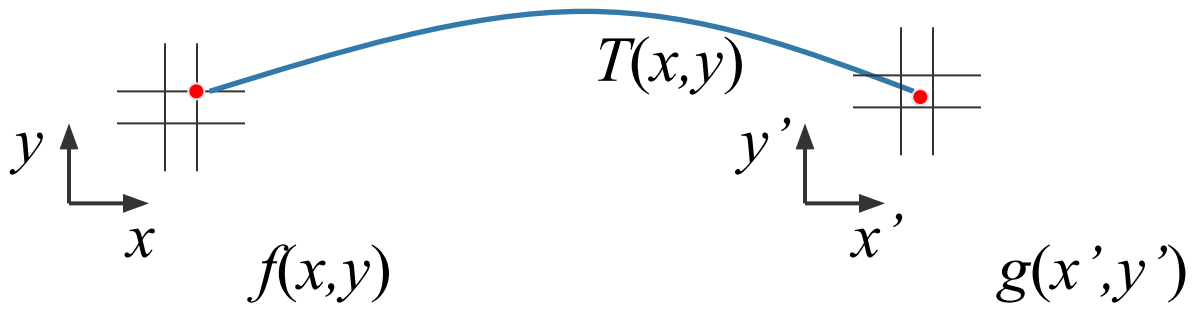
Forward warping

- Send each pixel $f(x,y)$ to its corresponding location
- $(x',y') = T(x,y)$ in the second image
- What if pixel lands "between" two pixels?



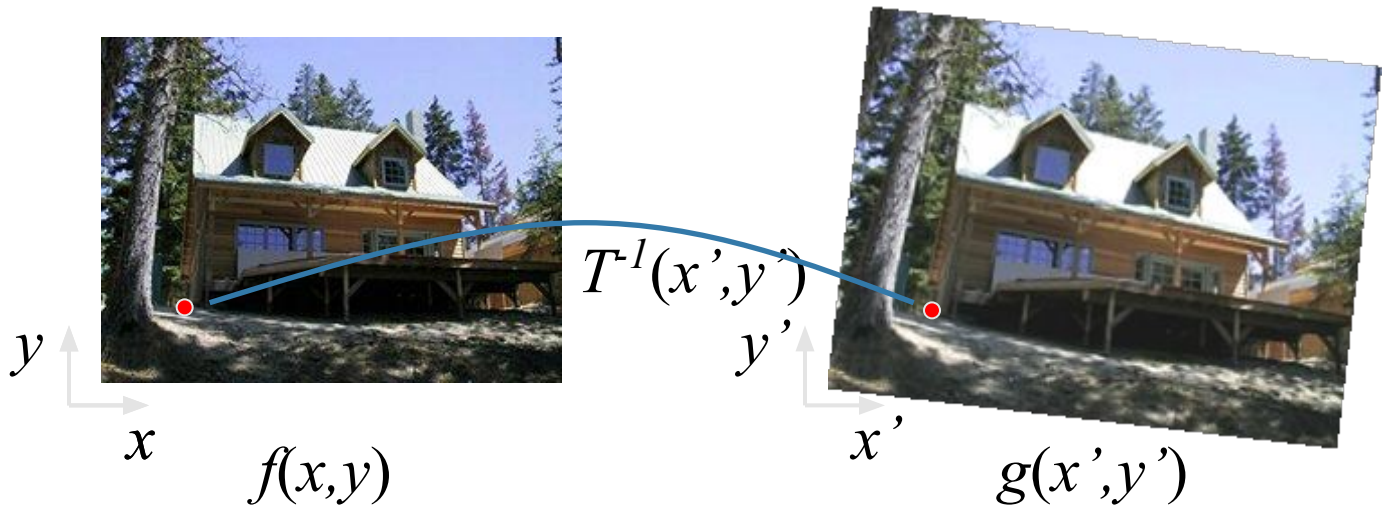
Forward warping

- Send each pixel $f(x,y)$ to its corresponding location
- $(x',y') = T(x,y)$ in the second image
- What if pixel lands "between" two pixels?
  - Distribute color among neighboring pixels (x',y')
    - Known as "splatting"



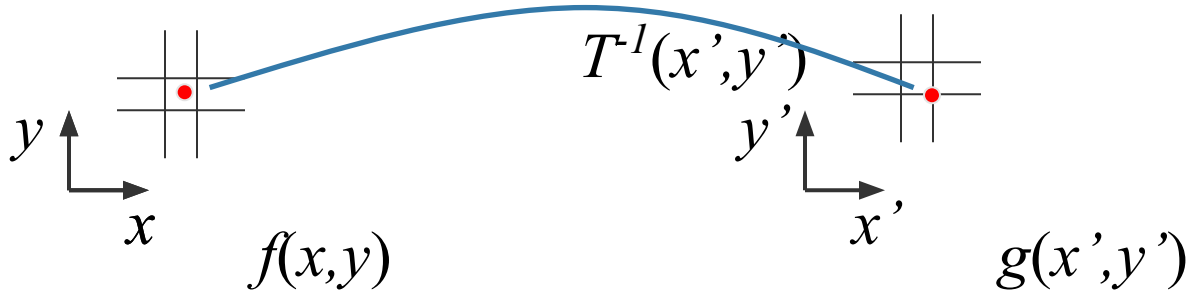
Inverse warping

- Get each pixel $g(x',y')$ from its corresponding location
- $(x,y) = T^{-1}(x',y')$ in the first image
- What if pixel comes from "between" two pixels?



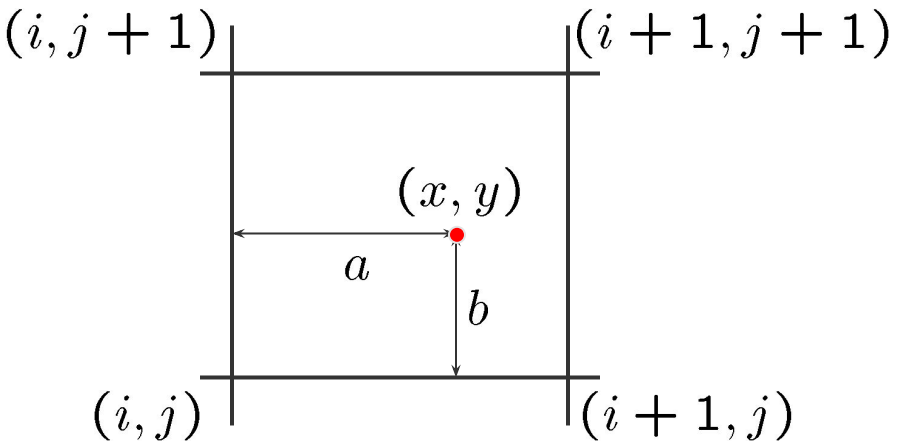
Inverse warping

- Get each pixel $g(x',y')$ from its corresponding location
- $(x,y) = T^{-1}(x',y')$ in the first image
- What if pixel comes from "between" two pixels?
- Interpolate color value from neighbors
  - nearest neighbor, bilinear, Gaussian, bicubic



Bilinear interpolation

- A simple method for resampling images



$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

Forward vs. inverse warping

- Which is better?
- Usually inverse; eliminates holes
  - However, it requires an invertible warp function—not always possible...