# Image Warping

Computational Photography

Derek Hoiem, University of Illinois

Many slides from Alyosha Efros + Steve Seitz

Photo by Sean Carroll

# Reminder: Proj 2 due monday

- Much more difficult than project 1 – get started asap if not already
- Must compute SSD cost for every pixel (slow but not horribly slow using filtering method; see tips at end of project page)
- Learn how to debug visual algorithms: imshow, plot, dbstop if error, keyboard and break points are your friends
  - Suggestion: For “quilt_simple”, first set upper-left patch to be upper-left patch in source and iteratively find minimum cost patch and overlay --- should reproduce original source image, at least for part of the output

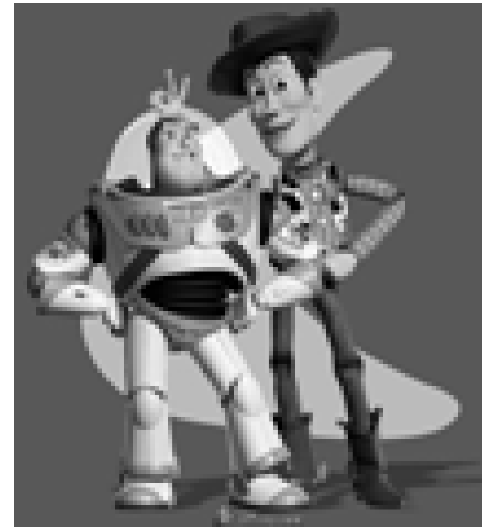
# Review from last class: Gradient Domain Editing

General concept: Solve for pixels of new image that satisfy constraints on the gradient and the intensity

- Constraints can be from one image (for filtering) or more (for blending)

# Project 3: Reconstruction from Gradients

1. Preserve x-y gradients
2. Preserve intensity of one pixel



Source pixels: s

Variable pixels: v

1. minimize (v(x+1,y)-v(x,y) - (s(x+1,y)-s(x,y))^2
2. minimize (v(x,y+1)-v(x,y) - (s(x,y+1)-s(x,y))^2
3. minimize (v(1,1)-s(1,1))^2

# Project 3 (extra): NPR

- Preserve gradients on edges
  - e.g., get canny edges with edge(im, 'canny')
- Reduce gradients not on edges
- Preserve original intensity

Perez et al. 2003

# Colorization using optimization

- Solve for uv channels (in Luv space) such that similar intensities have similar colors

- Minimize squared color difference, weighted by intensity similarity

$$J(U) = \sum_{\mathbf{r}} \left( U(\mathbf{r}) - \sum_{\mathbf{s} \in N(\mathbf{r})} w_{\mathbf{rs}} U(\mathbf{s}) \right)^2$$

- Solve with sparse linear system of equations

http://www.cs.huji.ac.il/~yweiss/Colorization/

# Gradient-domain editing

Many image processing applications can be thought of as trying to manipulate gradients or intensities:

- Contrast enhancement
- Denoising
- Poisson blending
- HDR to RGB
- Color to Gray
- Recoloring
- Texture transfer

See Perez et al. 2003 and GradientShop for many examples

# Gradient-domain processing

Saliency-based Sharpening

http://www.gradientshop.com

# Gradient-domain processing

Non-photorealistic rendering

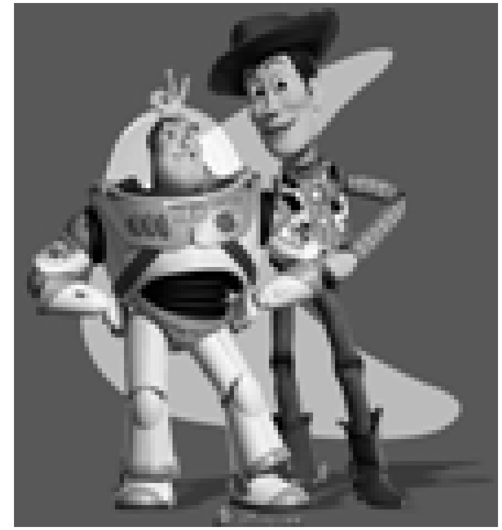
http://www.gradientshop.com

# Project 3: Gradient Domain Editing

General concept: Solve for pixels of new image that satisfy constraints on the gradient and the intensity

- Constraints can be from one image (for filtering) or more (for blending)

# Project 3: Reconstruction from Gradients



1. Preserve x-y gradients
2. Preserve intensity of one pixel

Source pixels: s

Variable pixels: v

1. minimize (v(x+1,y)-v(x,y) - (s(x+1,y)-s(x,y))^2
2. minimize (v(x,y+1)-v(x,y) - (s(x,y+1)-s(x,y))^2
3. minimize (v(1,1)-s(1,1))^2

# Project 3 (extra): NPR

- Preserve gradients on edges
  - e.g., get canny edges with edge(im, ‘canny’)
- Reduce gradients not on edges
- Preserve original intensity

Perez et al. 2003

# Take-home questions

1) I am trying to blend this bear into this pool. What problems will I have if I use:
   a) Alpha compositing with feathering
   b) Laplacian pyramid blending
   c) Poisson editing?



# Take-home questions

2) How would you make a sharpening filter using gradient domain processing? What are the constraints on the gradients and the intensities?

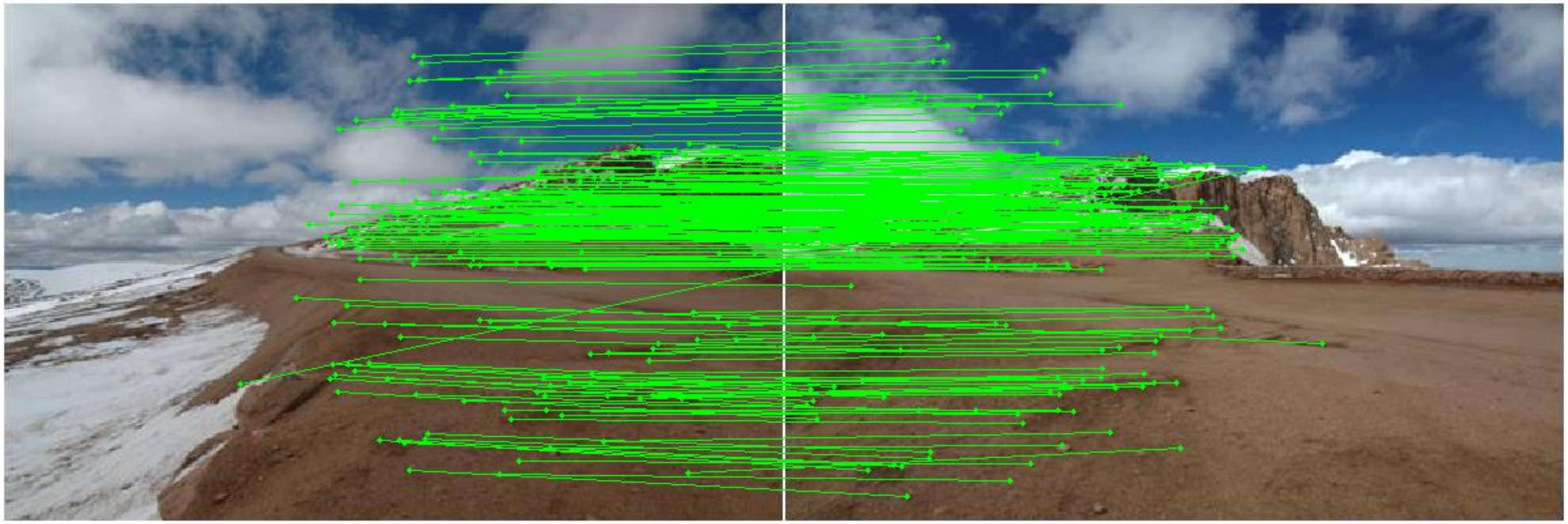
# Next two classes: warping and morphing

- Today
  - Global coordinate transformations
  - Image alignment

- Tuesday
  - Interpolation and texture mapping
  - Meshes and triangulation
  - Shape morphing

# Photo stitching: projective alignment

Find corresponding points in two images



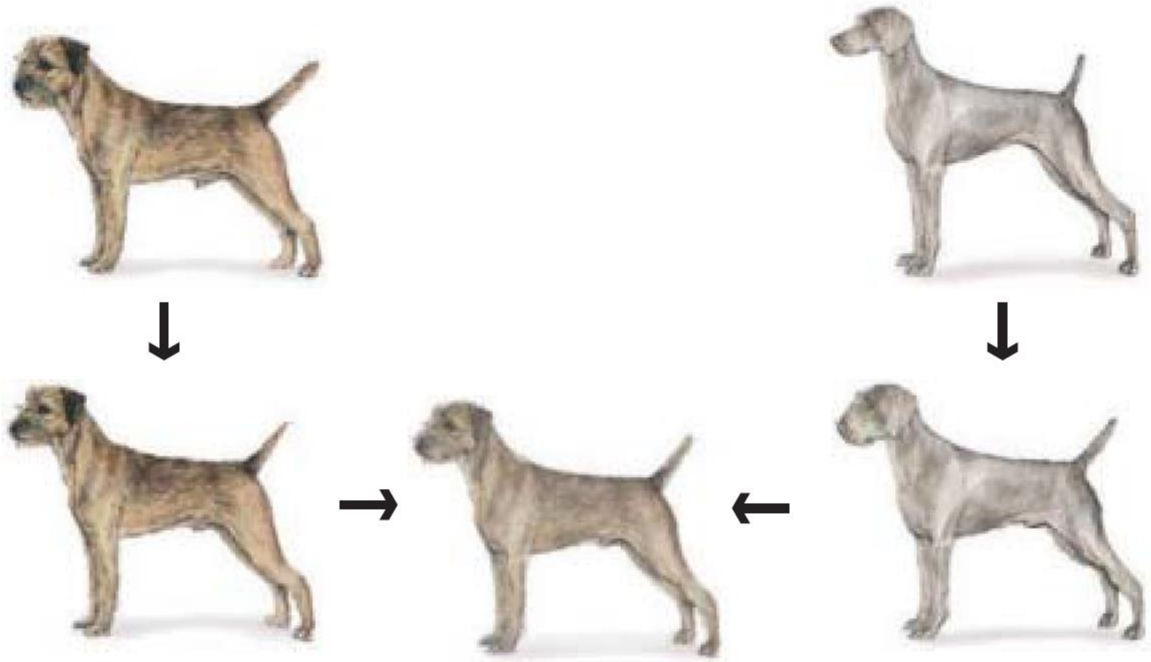
Solve for transformation that aligns the images

# Capturing light fields

Estimate light via projection from spherical surface onto image

# Morphing

Blend from one object to other with a series of local transformations



# Image Transformations

image filtering: change ***range*** of image

$$g(x) = T(f(x))$$

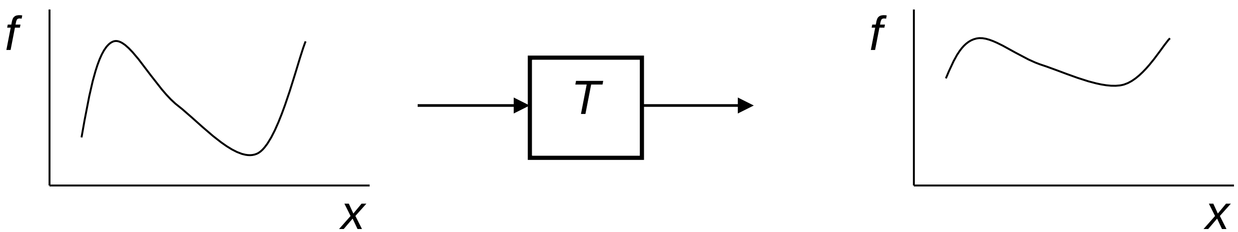
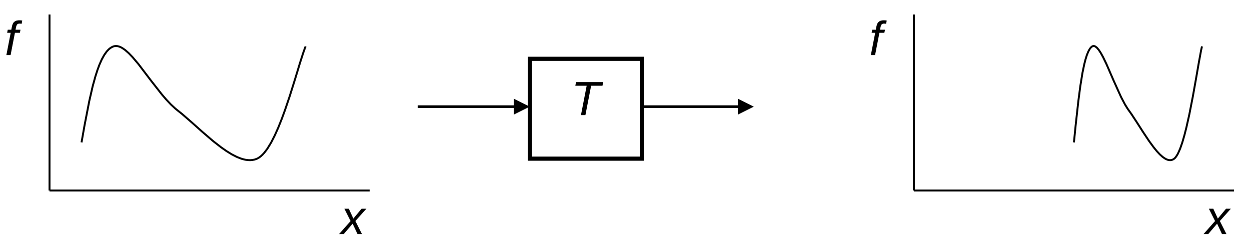



image warping: change ***domain*** of image

$$g(x) = f(T(x))$$



# Image Transformations

image filtering: change ***range*** of image

$$g(x) = T(f(x))$$

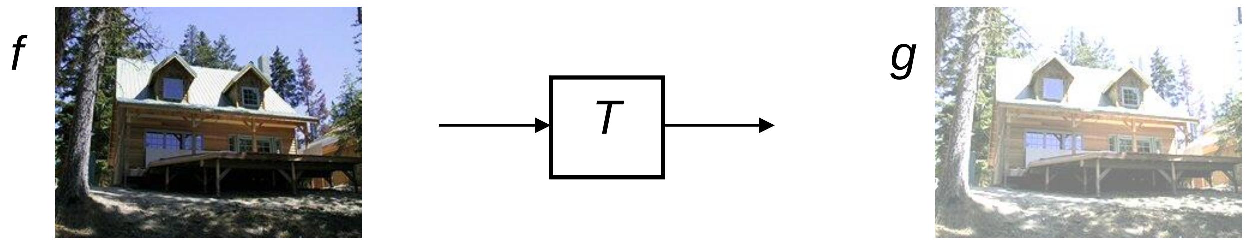
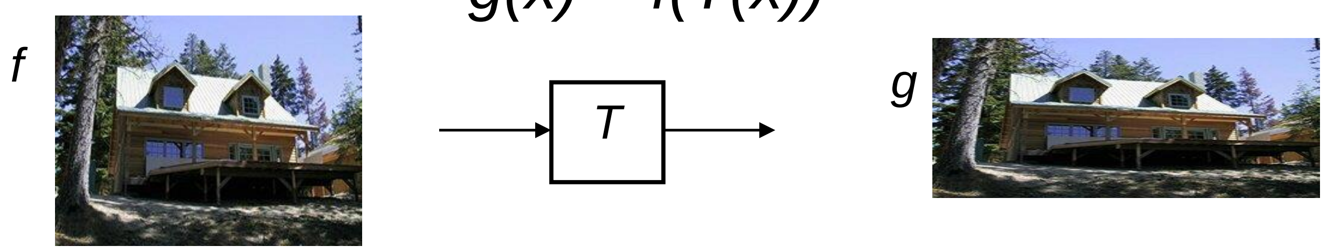
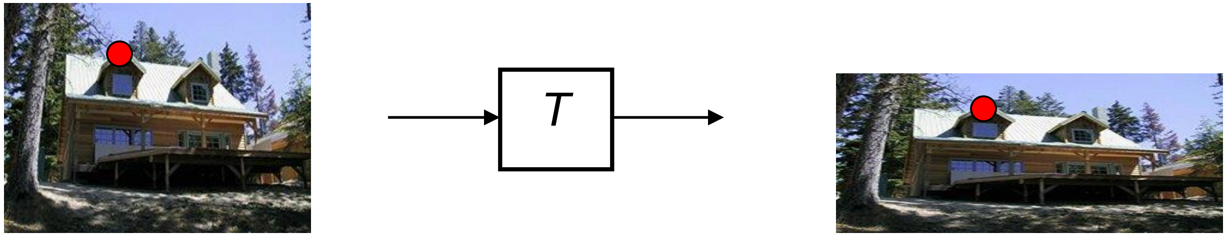



image warping: change ***domain*** of image

$$g(x) = f(T(x))$$



# Parametric (global) warping




**p** = (x,y) **p'** = (x',y')

Transformation T is a coordinate-changing machine:

p' = *T*(p)

What does it mean that *T* is global?

- Is the same for any point p
- can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

p' = **M**p

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

# Parametric (global) warping

## Examples of parametric warps:

translation

rotation

aspect

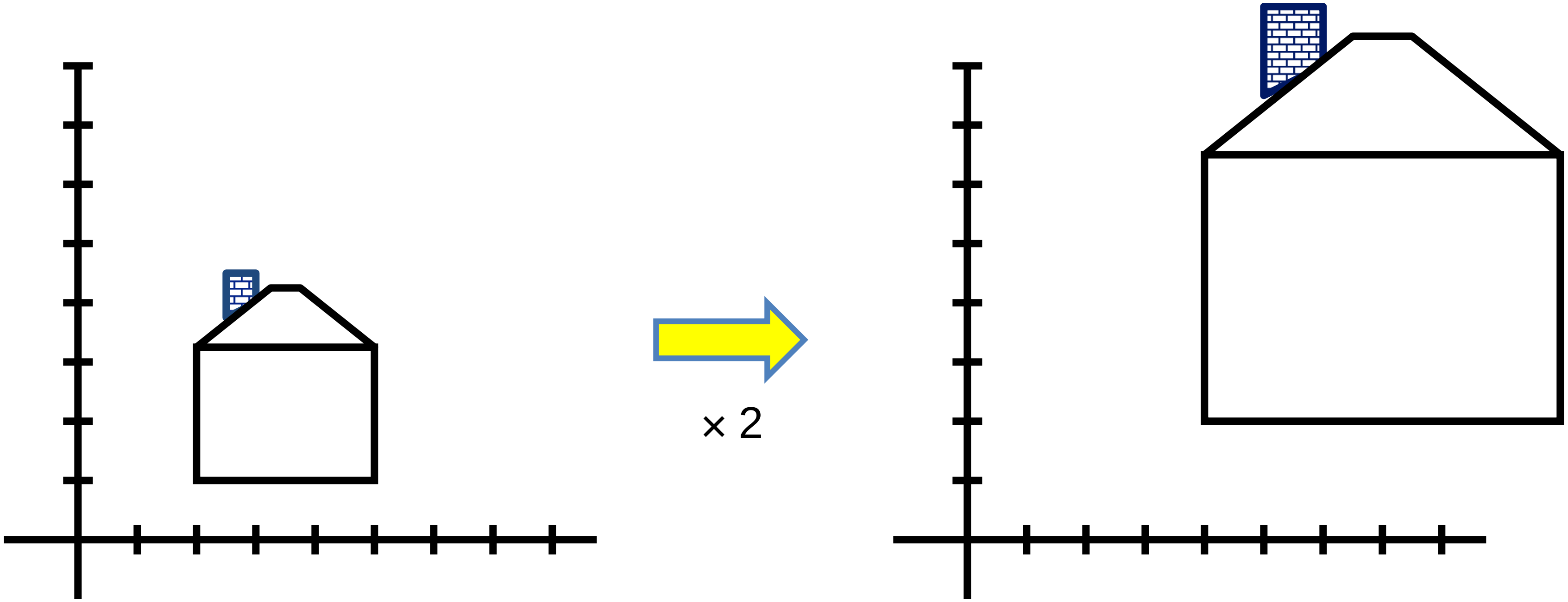
affine

perspective

cylindrical

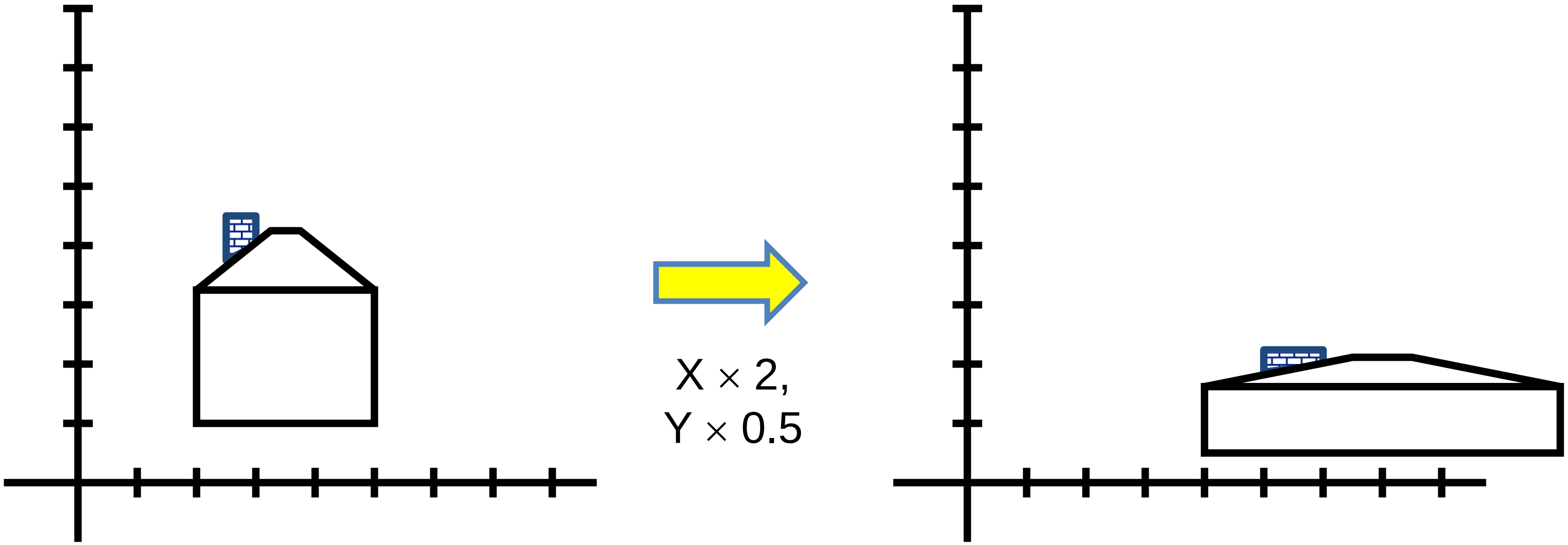
# Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



# Scaling

- *Non-uniform scaling*: different scalars per component:



# Scaling

- Scaling operation:

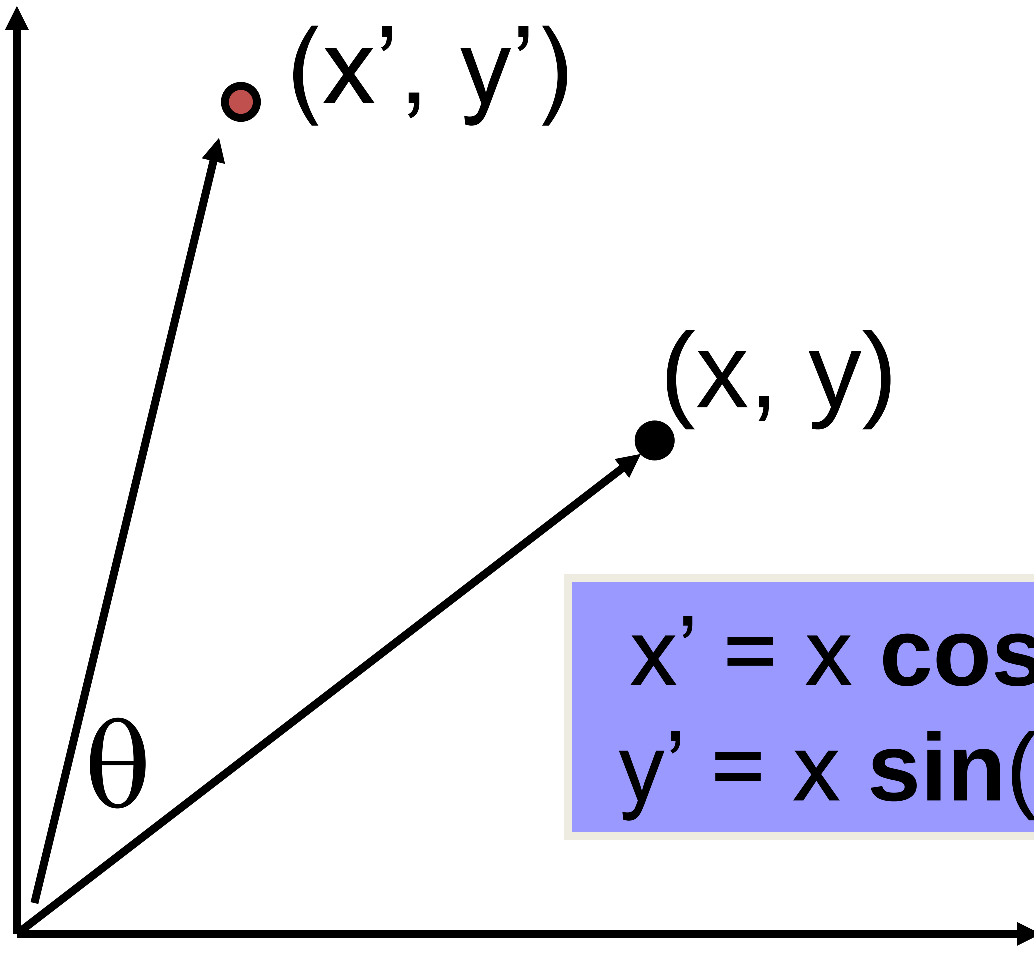
$$x' = ax$$
$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

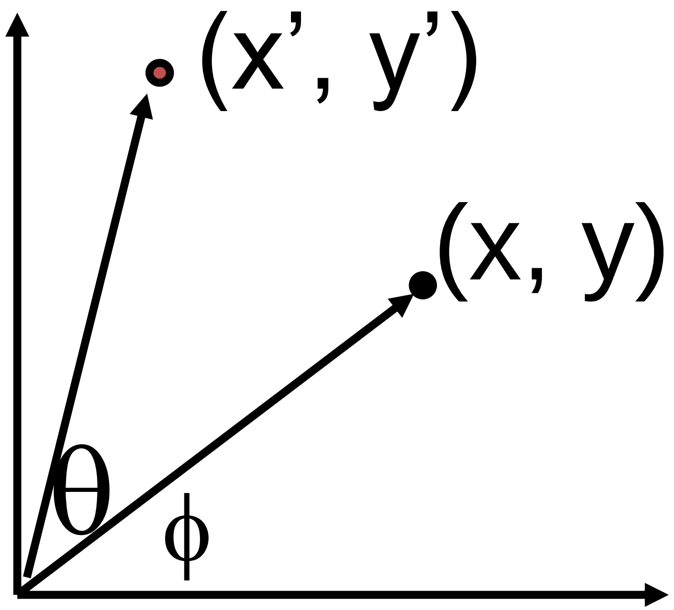
What is the transformation from (x', y') to (x, y)?

# 2-D Rotation




$$x' = x\,\mathbf{cos}(\theta) - y\,\mathbf{sin}(\theta)$$
$$y' = x\,\mathbf{sin}(\theta) + y\,\mathbf{cos}(\theta)$$

# 2-D Rotation




Polar coordinates…

$x = r \cos(\phi)$

$y = r \sin(\phi)$

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

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

Trig Identity…

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

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

Substitute…

$x' = x \mathbf{cos}(\theta) - y \mathbf{sin}(\theta)$

$y' = x \mathbf{sin}(\theta) + y \mathbf{cos}(\theta)$

# 2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

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 is the inverse transformation?

- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

# 2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

## 2D Identity?

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

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

## 2D Scale around (0,0)?

$$\boldsymbol{x' = s_x * x}$$
$$\boldsymbol{y' = s_y * y}$$

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

# 2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

## 2D Rotate around (0,0)?

$$x' = \cos\Theta * x - \sin\Theta * y$$
$$y' = \sin\Theta * x + \cos\Theta * y$$

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

## 2D Shear?

$$x' = x + k_x * y$$
$$y' = k_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k_x \\ k_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

# 2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

## 2D Mirror about Y axis?

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

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

## 2D Mirror over (0,0)?

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

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

# 2x2 Matrices

What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$x' = x + t_x$$

$$y' = y + t_y$$

NO!

# 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 are preserved
  - Closed under composition

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

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

# Homogeneous Coordinates

Q: How can we represent translation in matrix form?

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

# Homogeneous Coordinates

***Homogeneous coordinates***

- represent coordinates in 2 dimensions with a 3-vector

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{homogeneous coords}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# Homogeneous Coordinates

2D Points → Homogeneous Coordinates

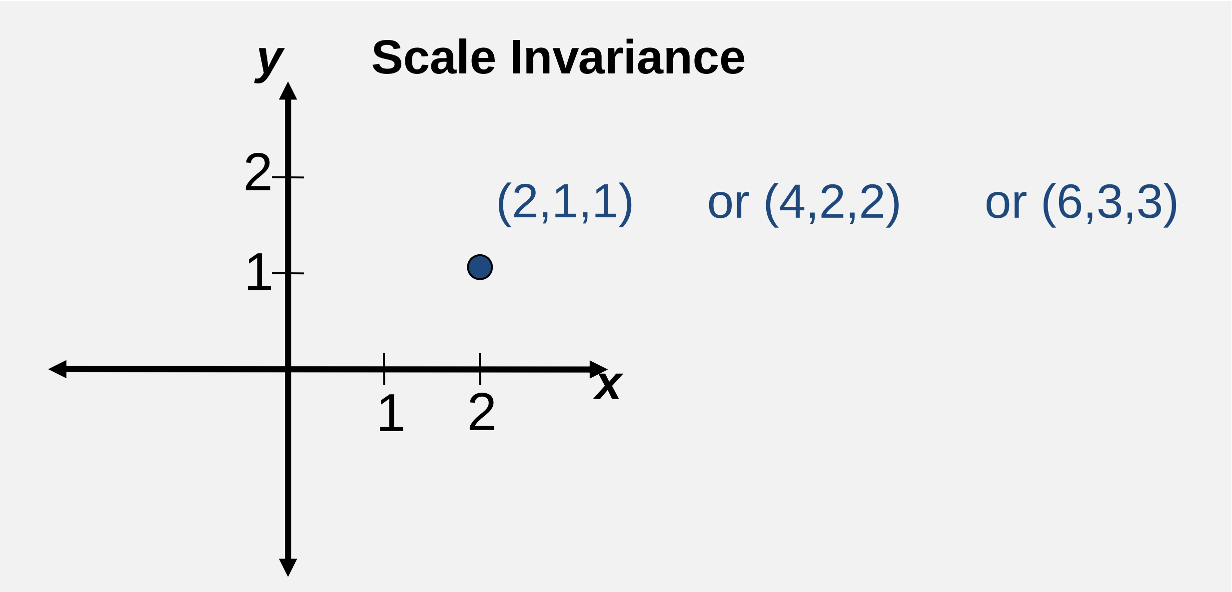
- Append 1 to every 2D point: (x y) → (x y 1)

Homogeneous coordinates → 2D Points

- Divide by third coordinate (x y w) → (x/w y/w)

Special properties

- Scale invariant: (x y w) = k * (x y w)
- (x, y, 0) represents a point at infinity
- (0, 0, 0) is not allowed



# Homogeneous Coordinates

Q: How can we represent translation in matrix form?

$$x' = x + t_x$$

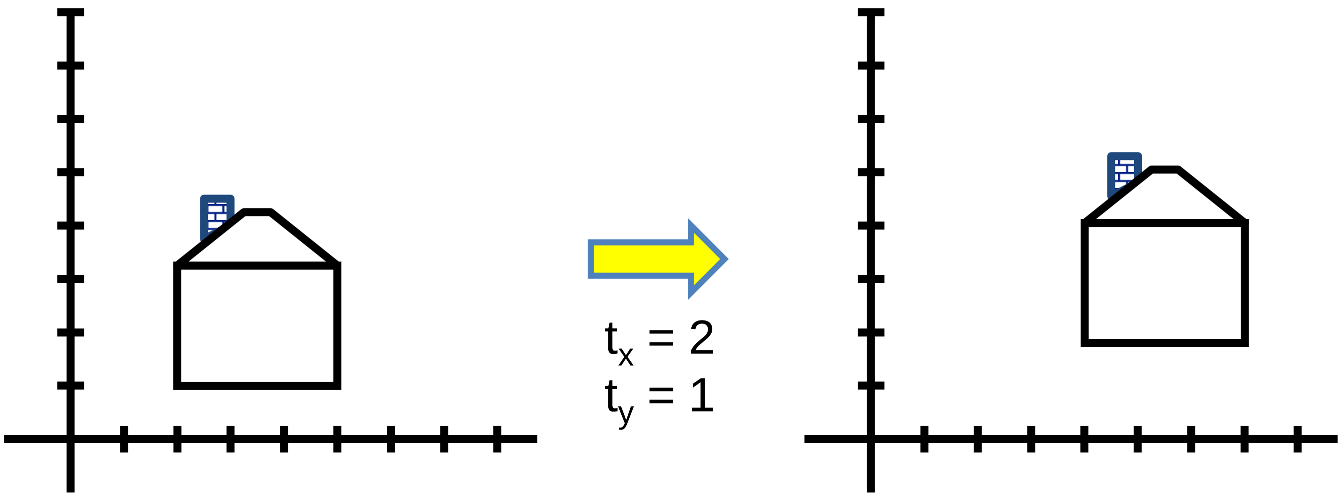
$$y' = y + t_y$$

A: Using the rightmost column:

$$\mathbf{Translation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

# Translation Example

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



# Basic 2D transformations as 3x3 matrices

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

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

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

Rotate

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

Shear

# Matrix Composition

Transformations can be combined by matrix multiplication

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

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

Does the order of multiplication matter?

# Affine Transformations

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

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Origin does not necessarily map to origin
- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

# Projective Transformations

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

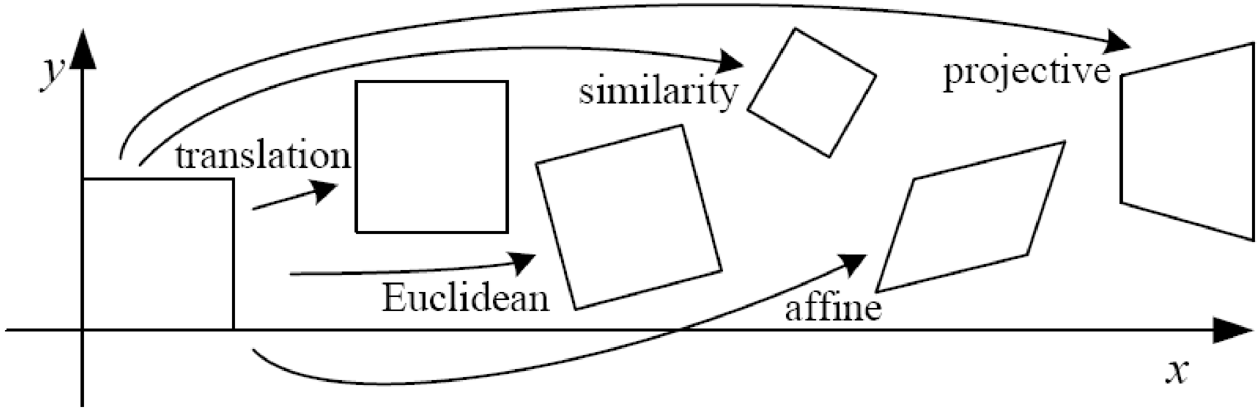
Projective transformations are combos of

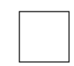
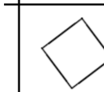
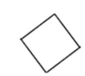
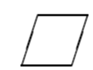
- Affine transformations, and
- Projective warps

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
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

# 2D image transformations

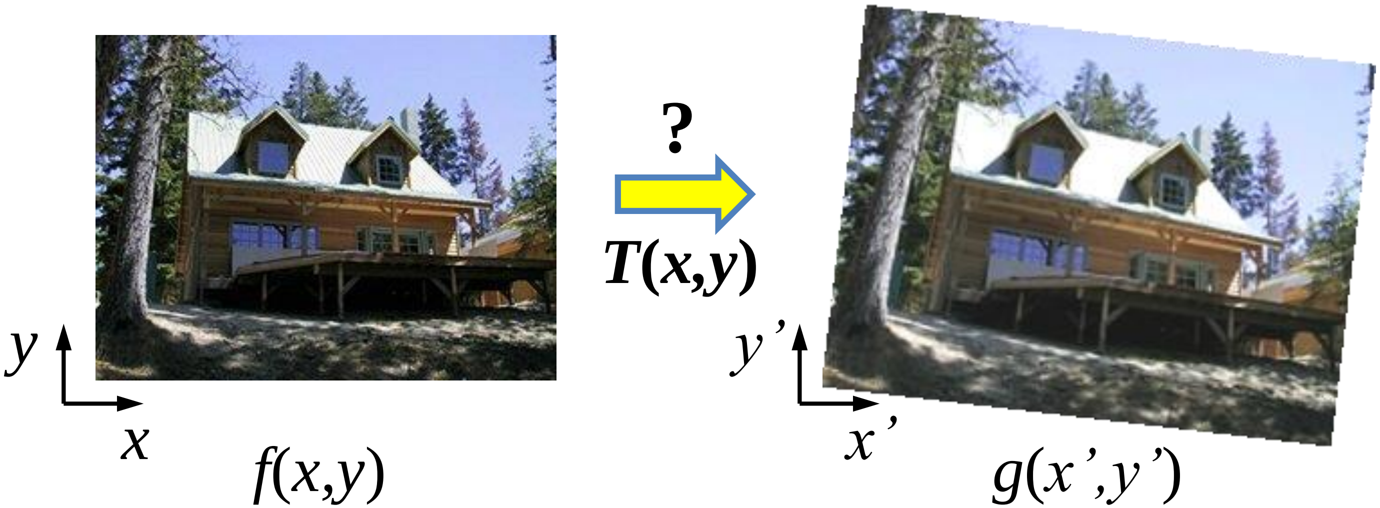



| Name | Matrix | # D.O.F. | Preserves: | Icon |
|---|---|---|---|---|
| translation | $\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$ | | |  |
| rigid (Euclidean) | $\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$ | | |  |
| similarity | $\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$ | | |  |
| affine | $\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$ | | |  |
| projective | $\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$ | | |  |

These transformations are a nested set of groups

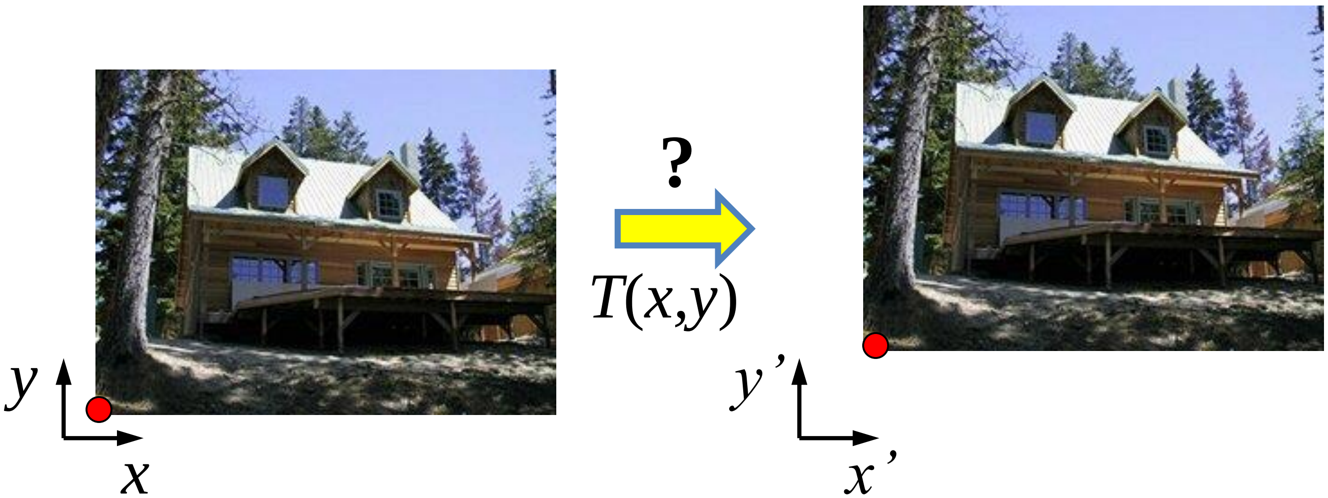
- Closed under composition and inverse is a member

# Recovering Transformations




- What if we know *f* and *g* and want to recover the transform T?
  - willing to let user provide correspondences
    - How many do we need?

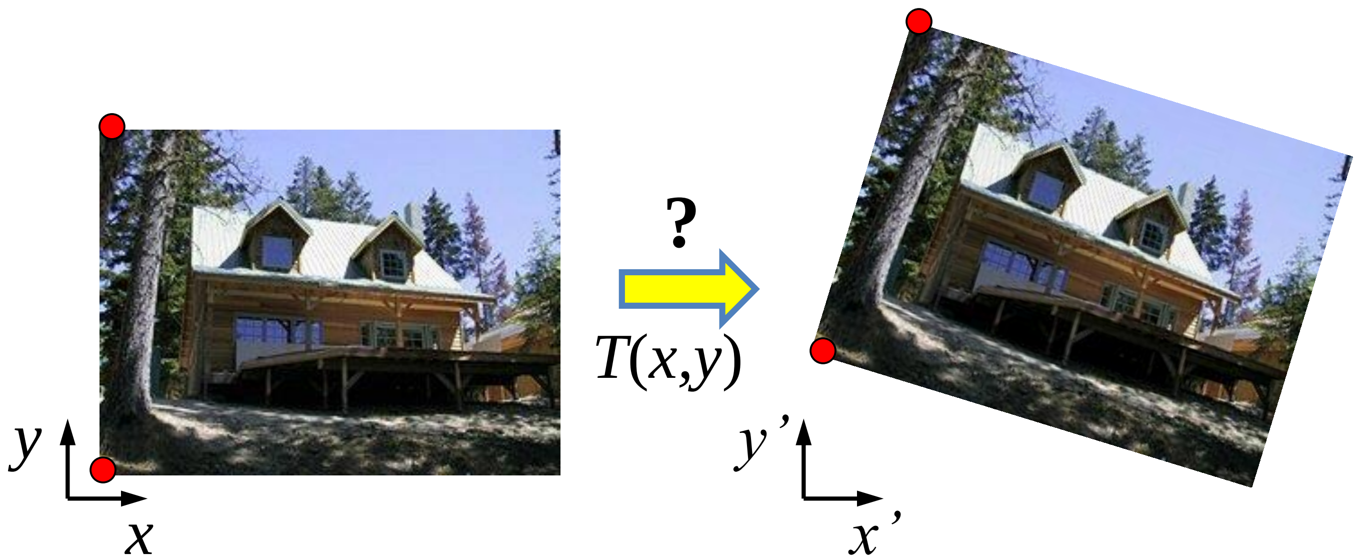
# Translation: # correspondences?




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

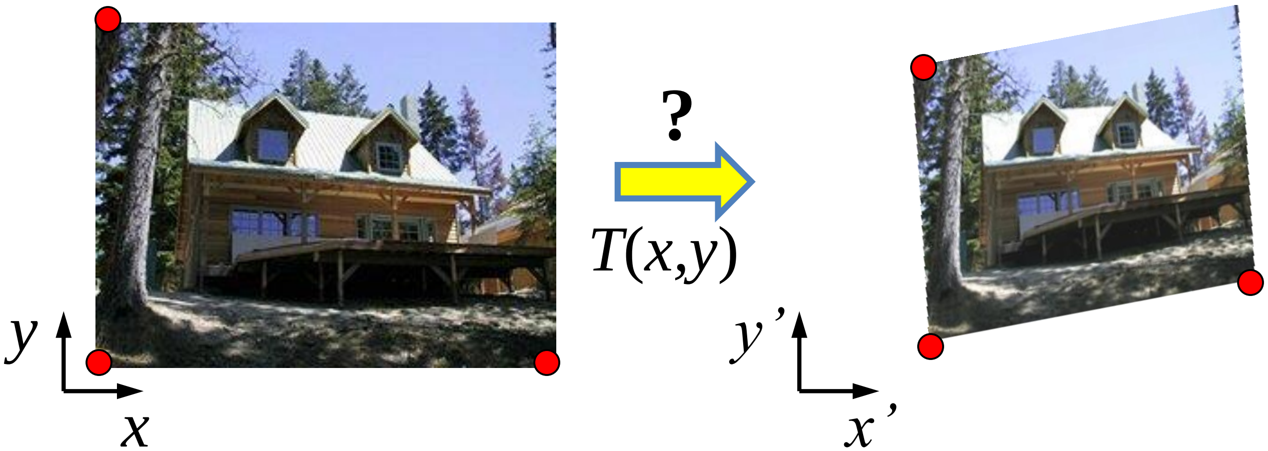
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & p'_x - p_x \\ 0 & 1 & p'_y - p_y \\ 0 & 0 & 1 \end{bmatrix}$$

# Euclidian: # correspondences?




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

# Affine: # correspondences?

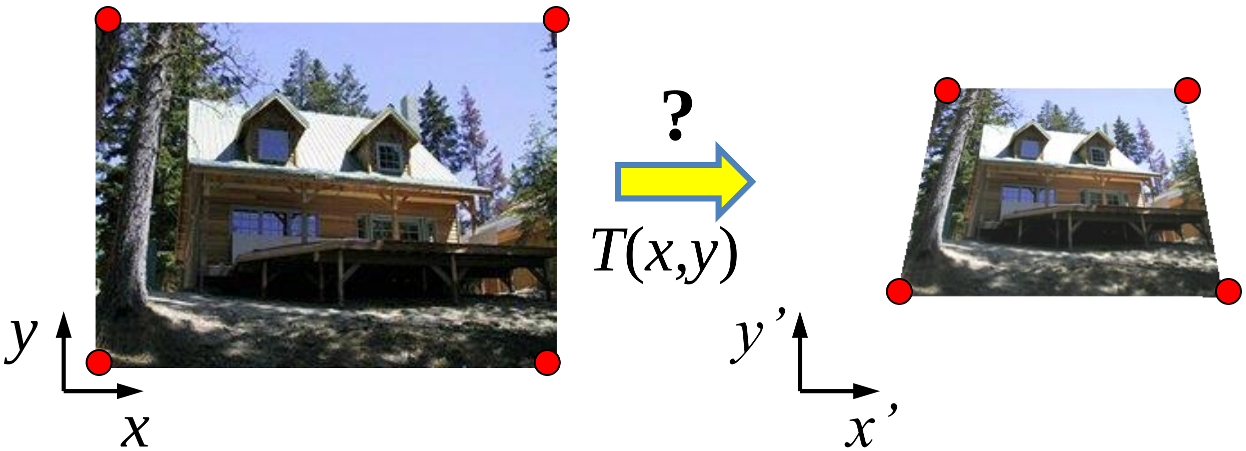



- How many DOF?
- How many correspondences needed for affine?

# Affine transformation estimation

- Math
- Matlab demo

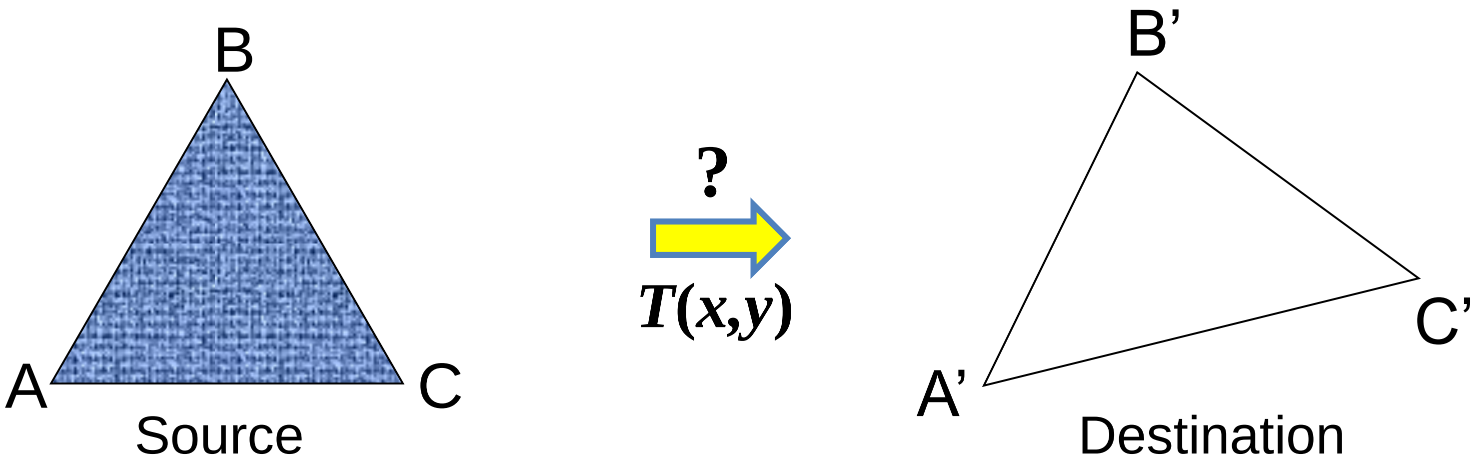
# Projective: # correspondences?




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

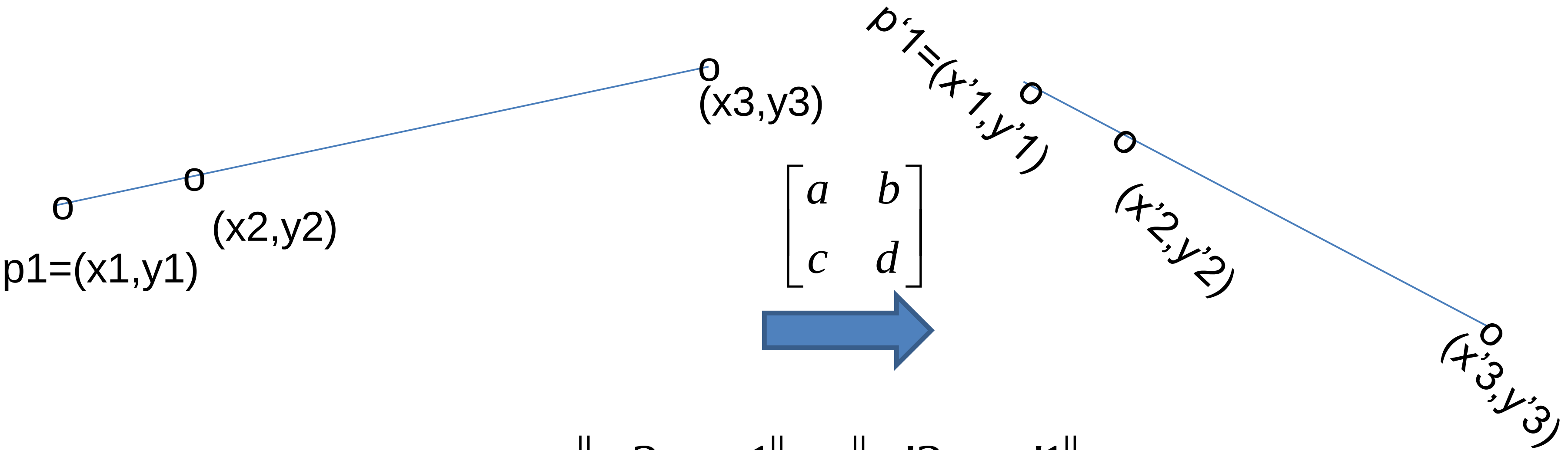
# Take-home Question

1) Suppose we have two triangles: ABC and A'B'C'. What transformation will map A to A', B to B', and C to C'? How can we get the parameters?



# Take-home Question

2) Show that distance ratios along a line are preserved under 2d linear transformations.




$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\frac{\|p2 - p1\|}{\|p3 - p1\|} = \frac{\|p'2 - p'1\|}{\|p'3 - p'1\|}$$

Hint: Write down x2 in terms of x1 and x3, given that the three points are co-linear

# Next class: texture mapping and morphing