2D transformations (a.k.a. warping)

Slide credits: Kris Kitani, Noah Snavely, Ioannis Gkioulekas

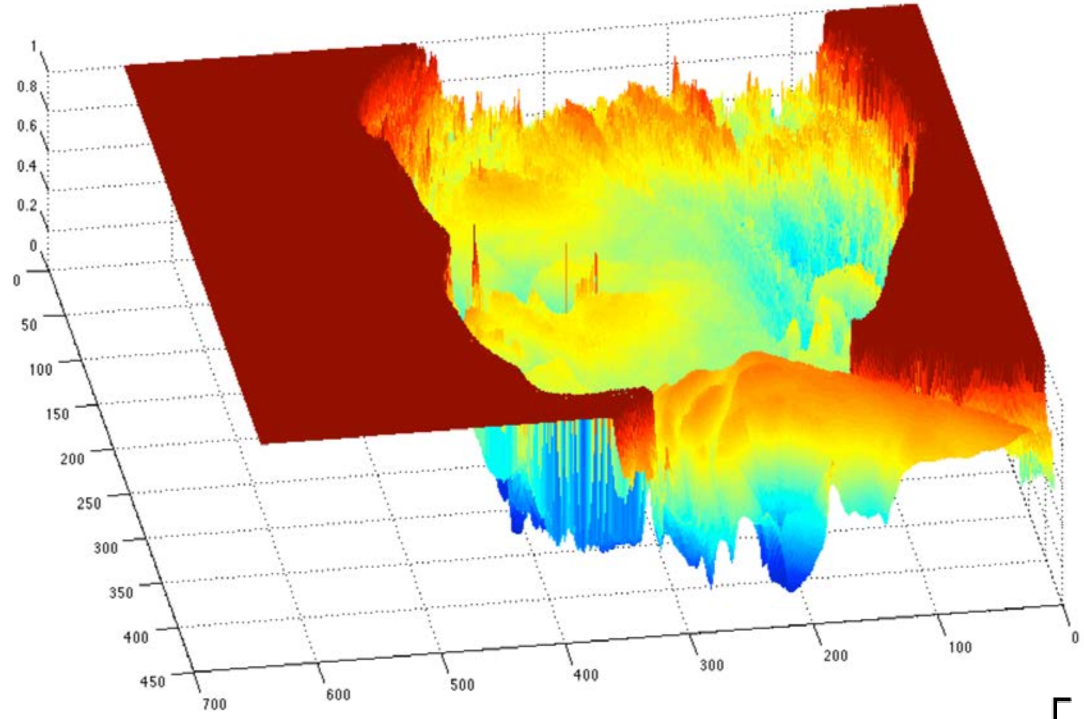
Reminder: image transformations

What is an image?

grayscale image

What is the range of the image function f?

$f(\boldsymbol{x})$



domain $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

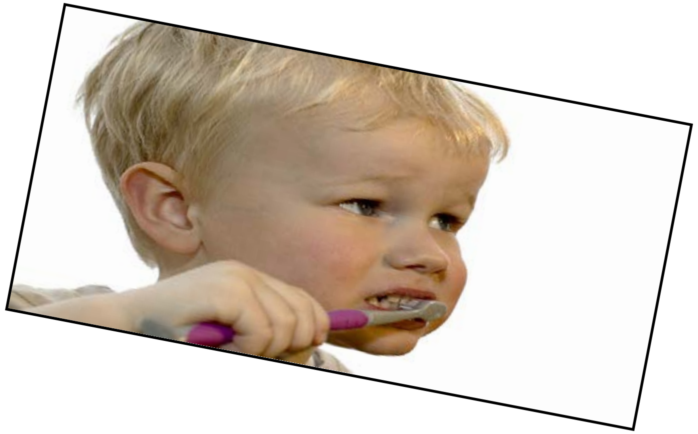
A (grayscale) image is a 2D function.

What types of image transformations can we do?

Filtering

changes pixel *values*

Warping



changes pixel *locations*

What types of image transformations can we do?

F

Filtering

$$G(\boldsymbol{x}) = h\{F(\boldsymbol{x})\}$$

G

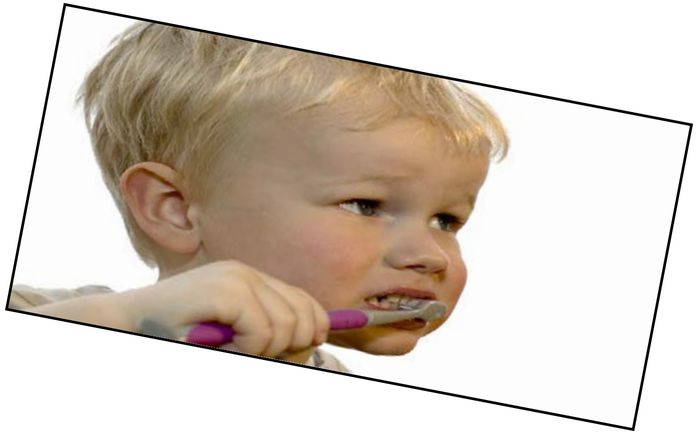
changes *range* of image function

F

Warping

$$G(\boldsymbol{x}) = F(h\{\boldsymbol{x}\})$$

G



changes *domain* of image function

Warping example: feature matching

Warping example: feature matching

Warping example: feature matching

- object recognition
- 3D reconstruction
- augmented reality
- image stitching

How do you compute the transformation?

Warping example: feature matching

Given a set of matched feature points:

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

point in one image

point in the other image

and a transformation:

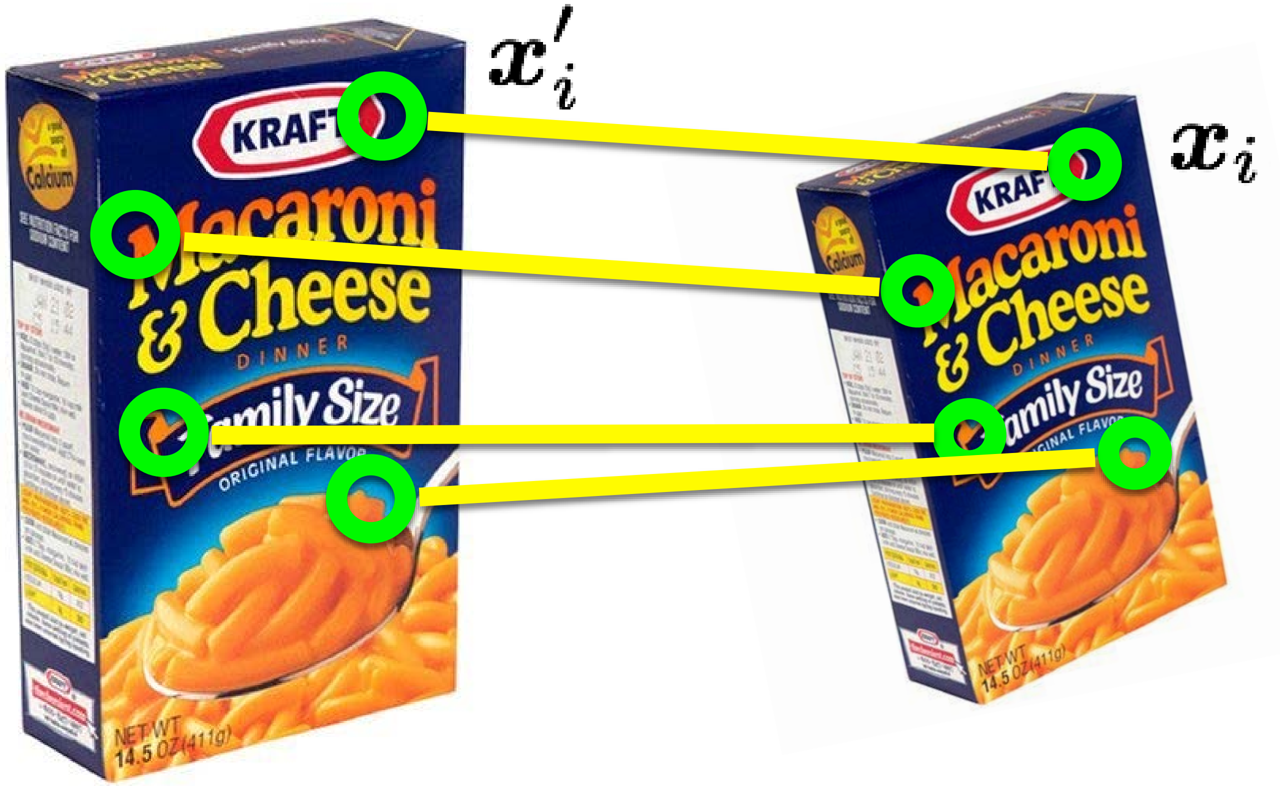
$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$

transformation function

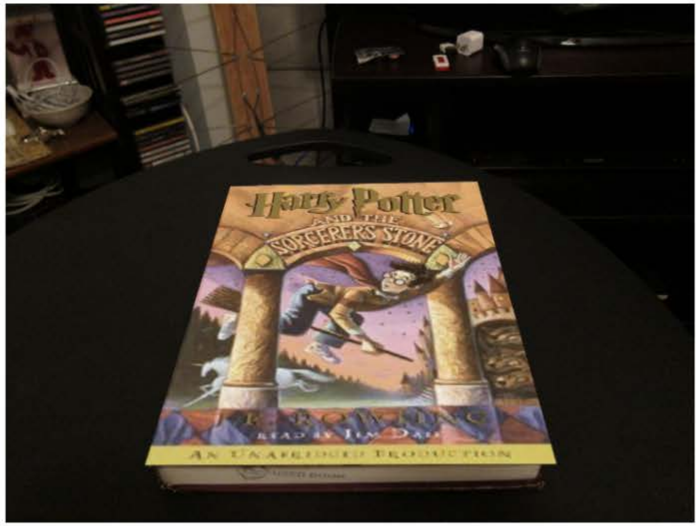
parameters

find the best estimate of the parameters

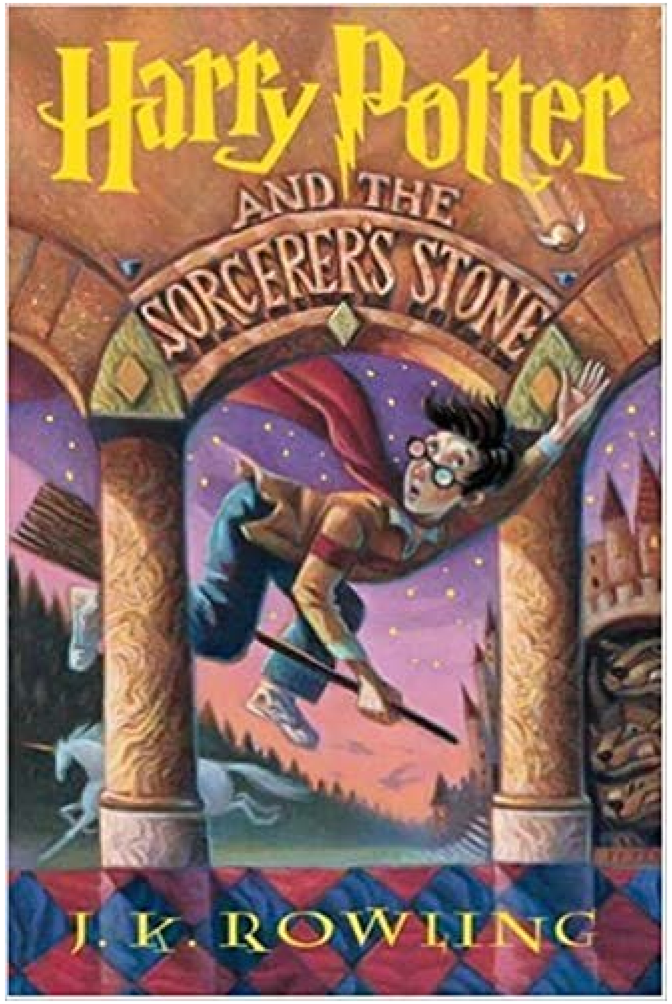
$$\boldsymbol{p}$$



What kind of transformation functions $\boldsymbol{f}$ are there?



Compute homography transformation



How do you create a panorama?

Panorama: an image of (near) 360^o field of view.

How do you create a panorama?

Panorama: an image of (near) 360^o field of view.

1. Use a very wide-angle lens.

Wide-angle lenses

Fish-eye lens: can produce (near) hemispherical field of view.

What are the pros and cons of this?

How do you create a panorama?

Panorama: an image of (near) 360^{o} field of view.

1. Use a very wide-angle lens.
- Pros: Everything is done optically, single capture.
- Cons: Lens is super expensive and bulky, lots of distortion (can be dealt-with in post).

Any alternative to this?

How do you create a panorama?

Panorama: an image of (near) 360^o field of view.

1. Use a very wide-angle lens.
- Pros: Everything is done optically, single capture.
- Cons: Lens is super expensive and bulky, lots of distortion (can be dealt-with in post).

2. Capture multiple images and combine them.

Panoramas from image stitching

1. Capture multiple images from different viewpoints.

2. Stitch them together into a virtual wide-angle image.

How do we stitch images from different viewpoints?

Idea 1: Translate one image relative to another.

How do we stitch images from different viewpoints?

Idea 1: Translate one image relative to another.

left on top

right on top

Translation-only stitching is not enough to mosaic these images.

How do we stitch images from different viewpoints?

What else can we try?

How do we stitch images from different viewpoints?

Use image homographies.

2D transformations

2D transformations

translation

rotation

aspect

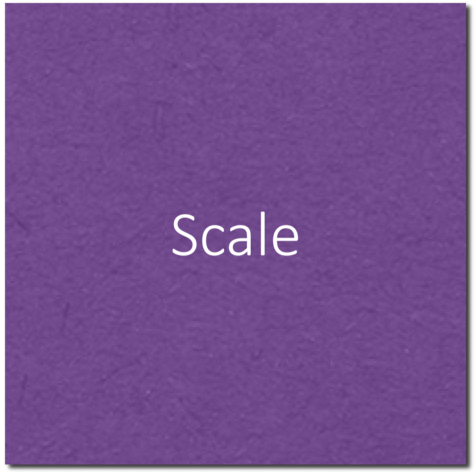
affine

perspective

cylindrical

2D planar transformations

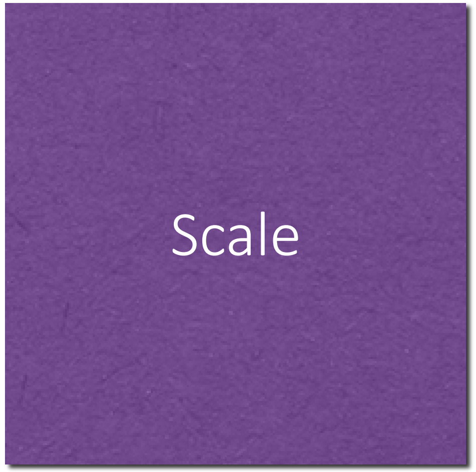
2D planar transformations



How would you implement scaling?

- Each component multiplied by a scalar
- Uniform scaling - same scalar for each component

2D planar transformations



y

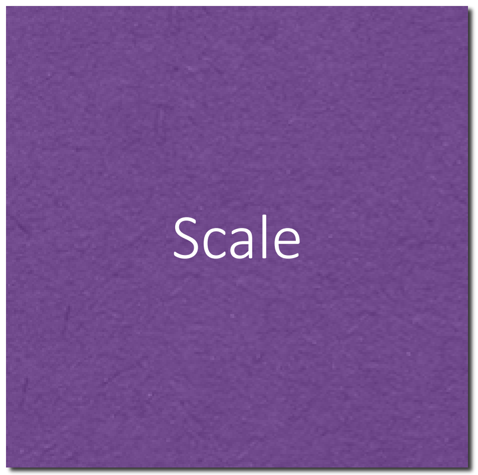
$$x' = ax$$
$$y' = by$$

What's the effect of using different scale factors?

- Each component multiplied by a scalar
- Uniform scaling - same scalar for each component

x

2D planar transformations



$$x' = ax$$

$$y' = by$$

matrix representation of scaling:

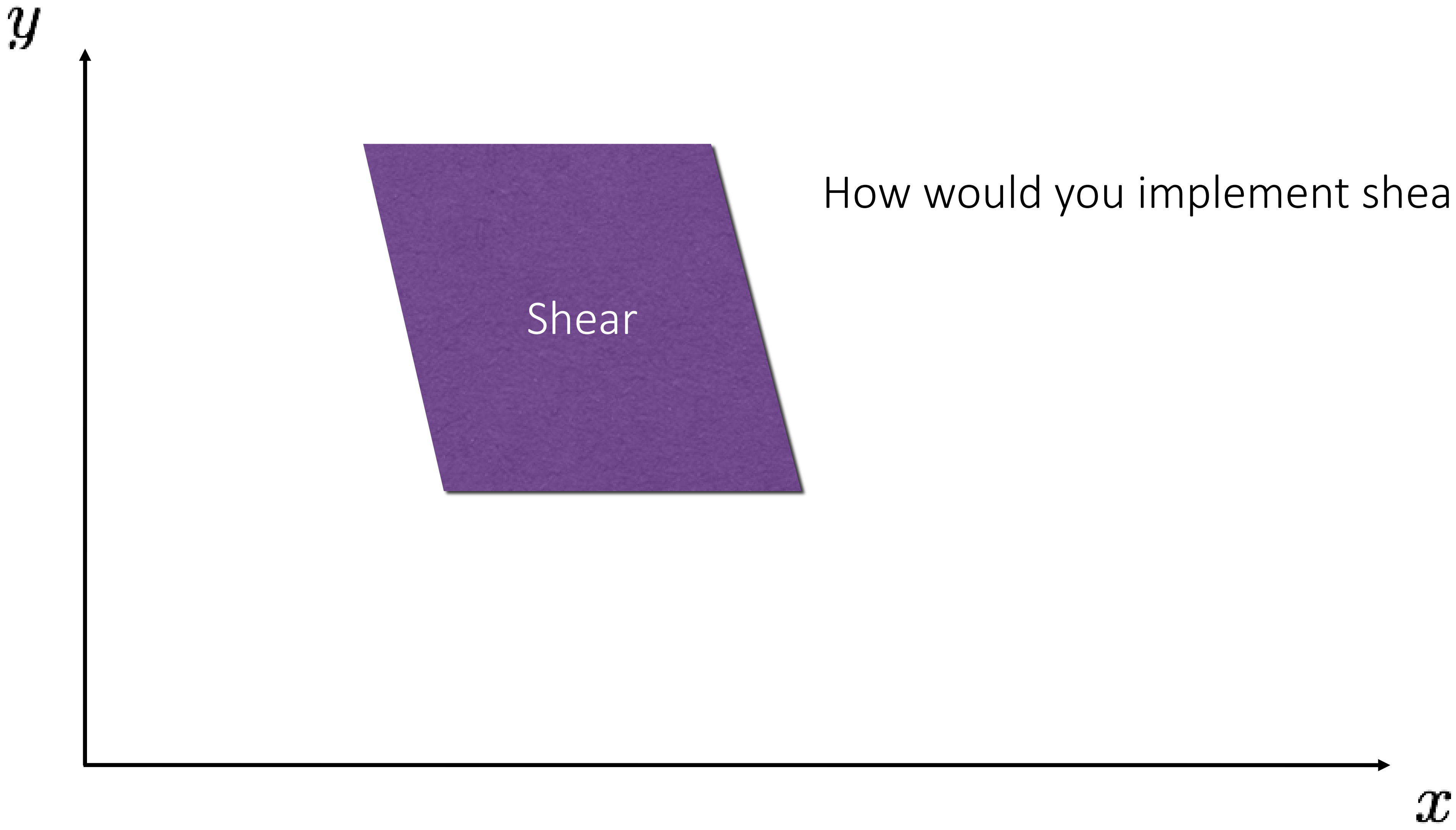
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix S}} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Each component multiplied by a scalar
- Uniform scaling - same scalar for each component

y

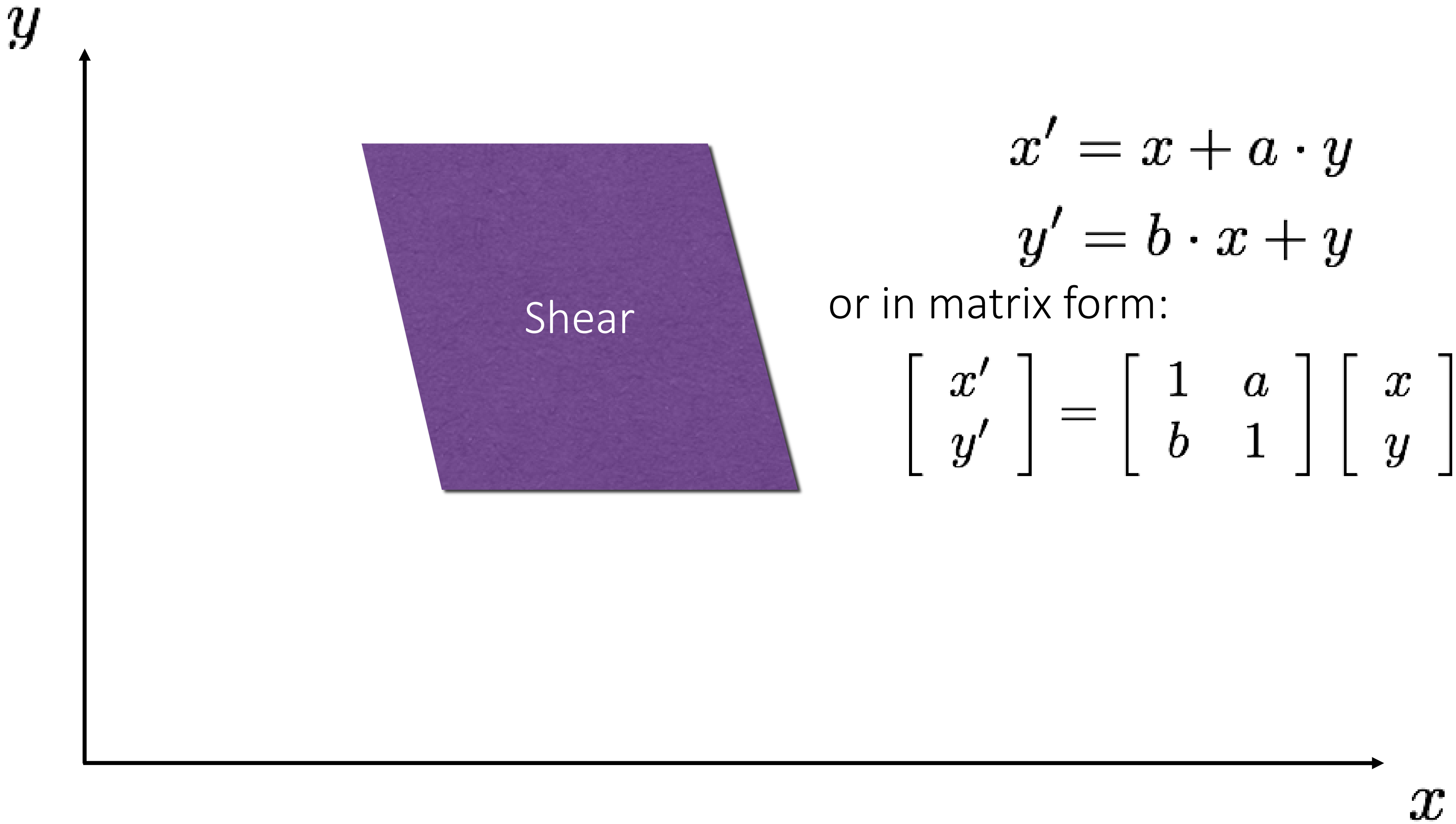
x

2D planar transformations



How would you implement shearing?

2D planar transformations



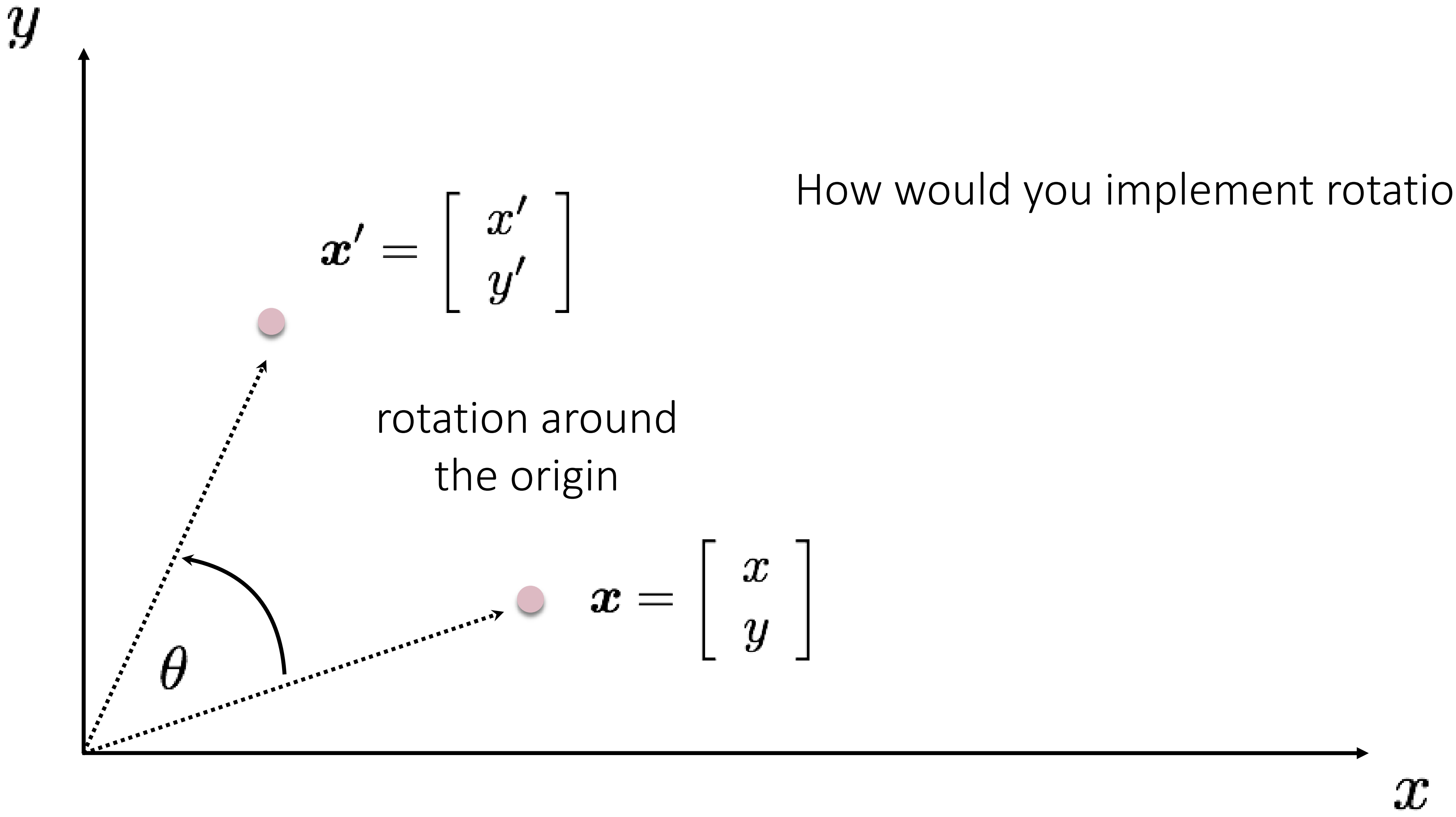
$$x' = x + a \cdot y$$

$$y' = b \cdot x + y$$

or in matrix form:

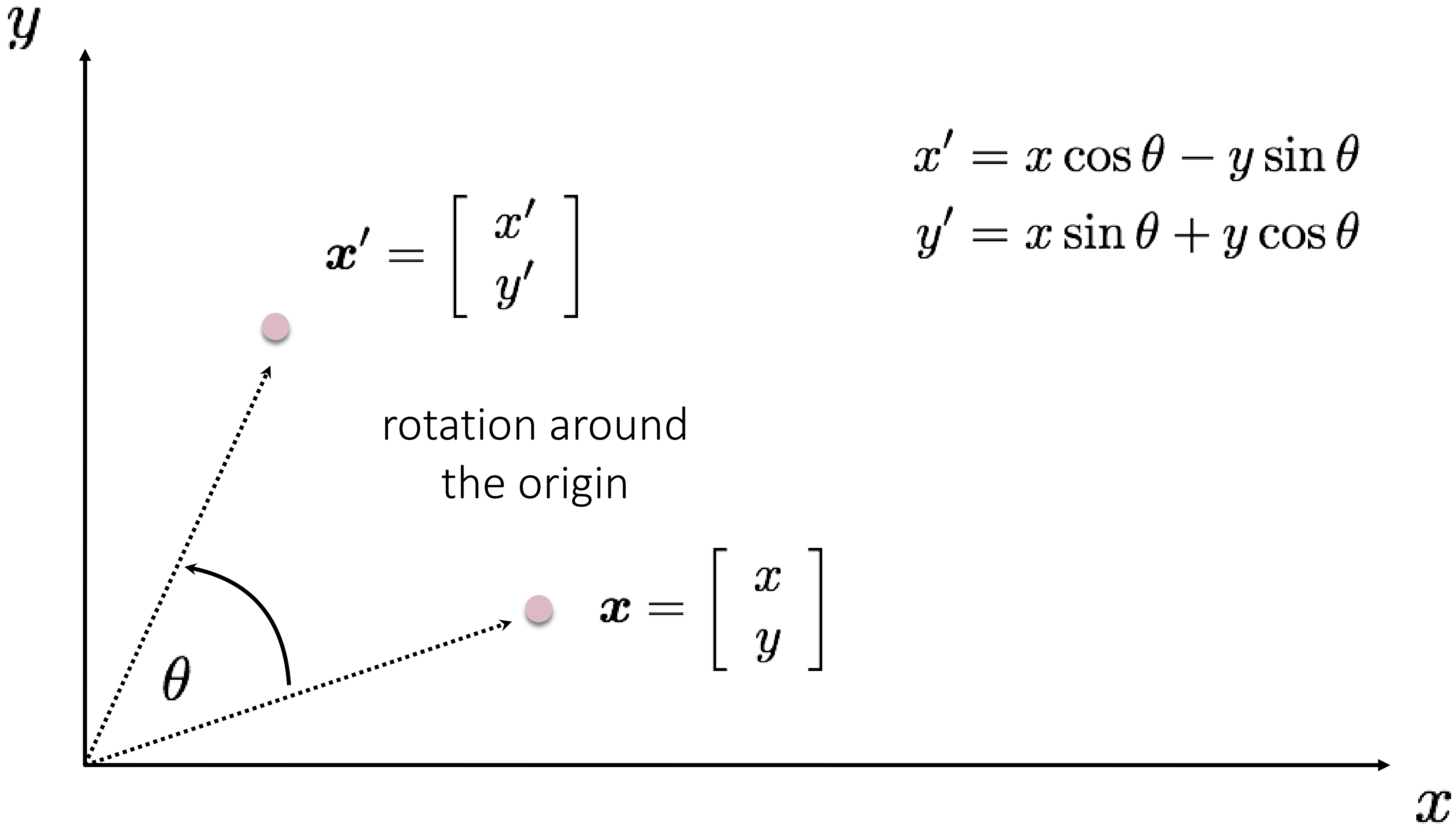
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D planar transformations



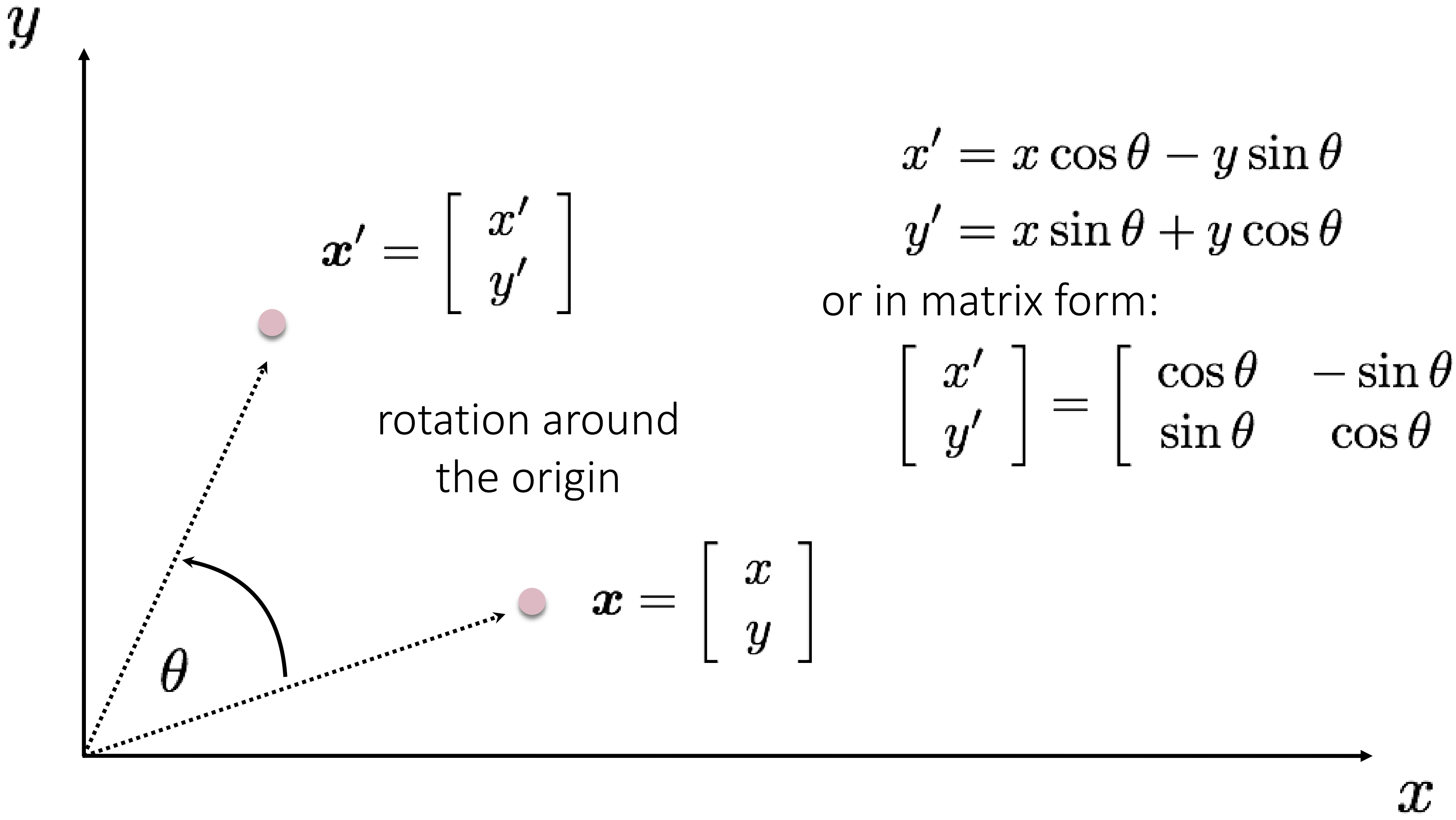
How would you implement rotation?

2D planar transformations



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

2D planar transformations



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

or in matrix form:

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

2D planar and linear transformations

$$\boldsymbol{x}' = f(\boldsymbol{x}; p)$$

$$\downarrow$$

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

parameters $\boldsymbol{p}$

point $\boldsymbol{x}$

2D planar and linear transformations

Scale

$$\mathbf{M} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Flip across y

$$\mathbf{M} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Rotate

$$\mathbf{M} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Flip across origin

$$\mathbf{M} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

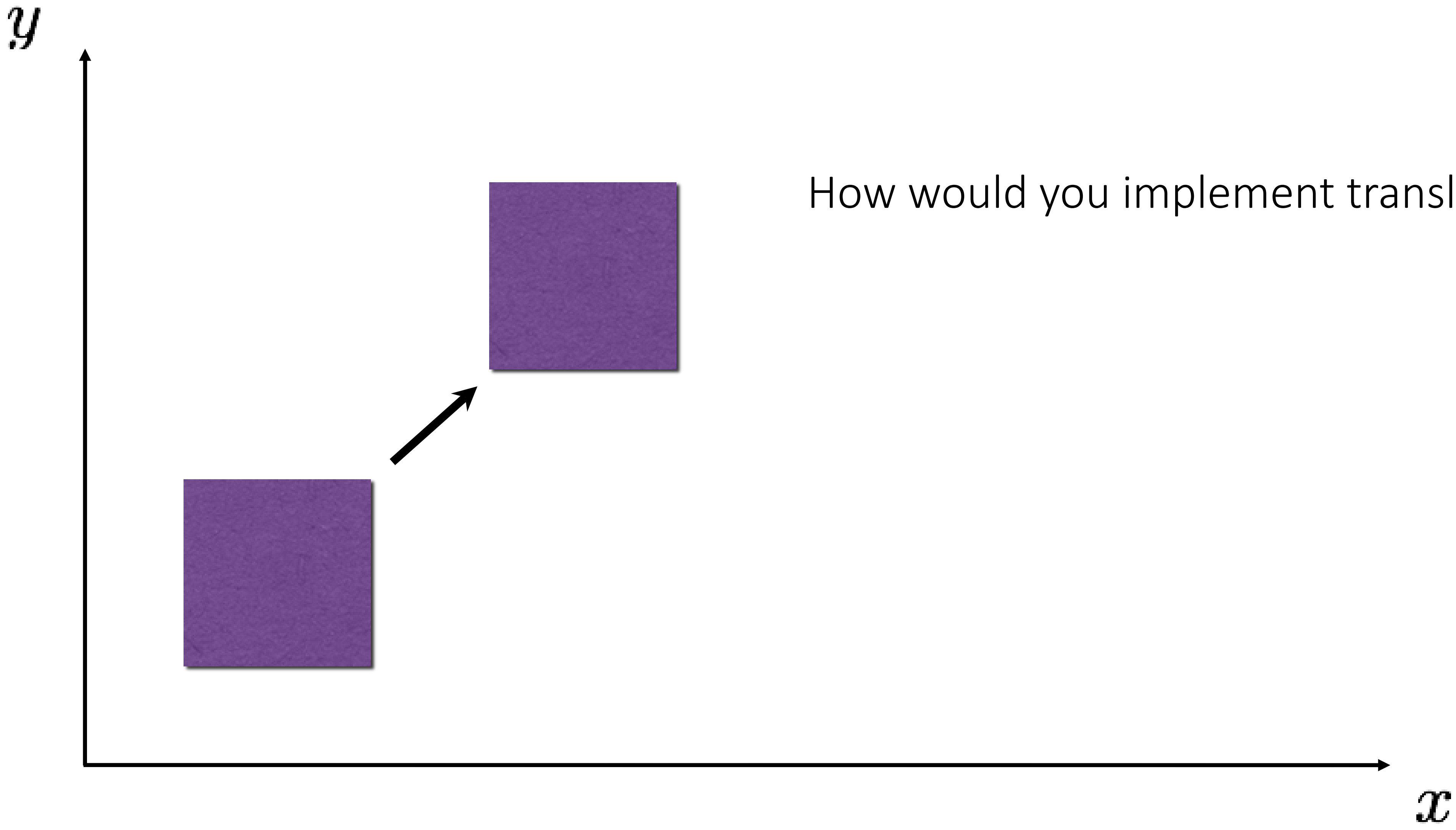
Shear

$$\mathbf{M} = \begin{bmatrix} 1 & s_x \\ s_y & 1 \end{bmatrix}$$

Identity

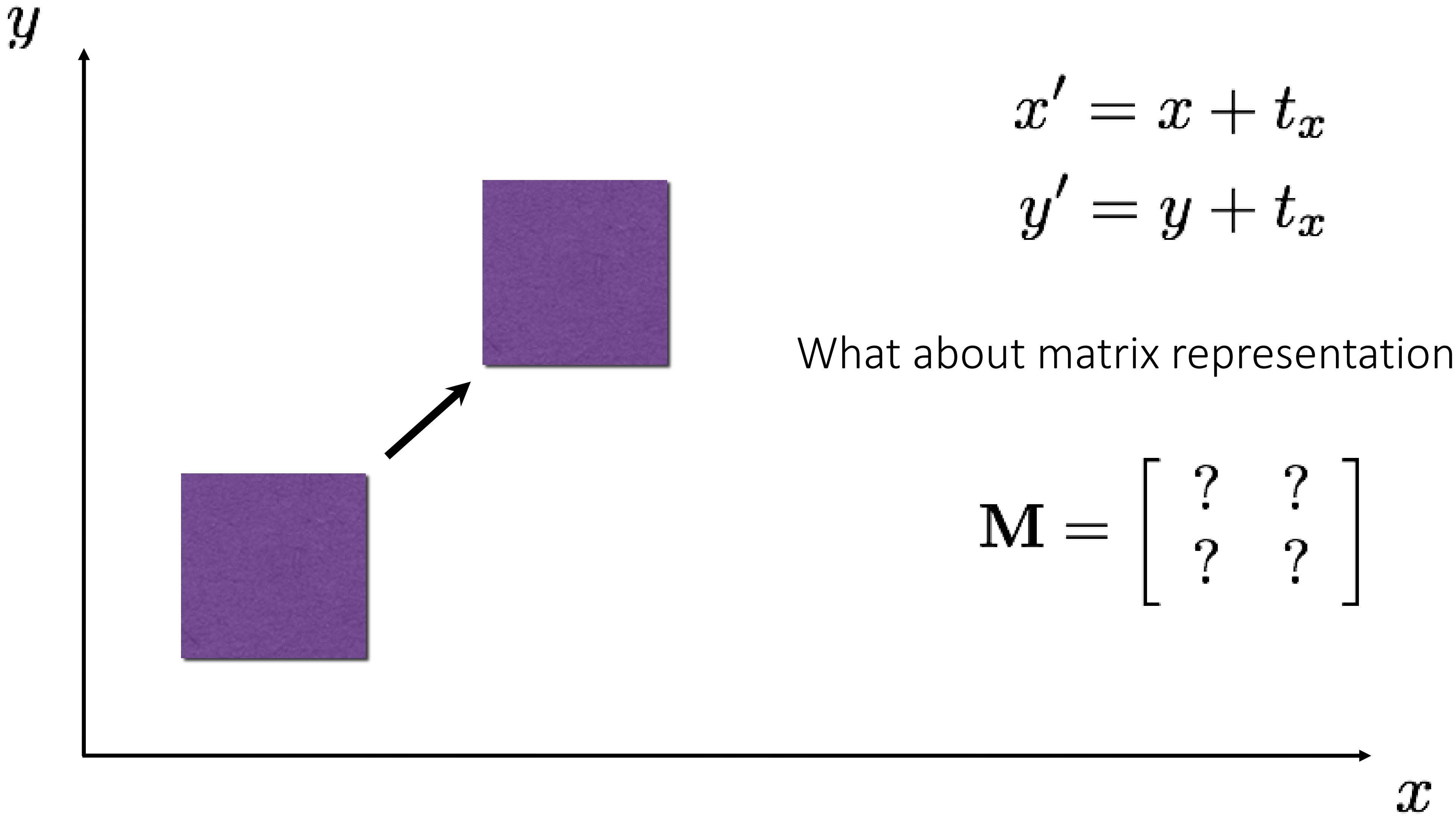
$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D translation



How would you implement translation?

2D translation

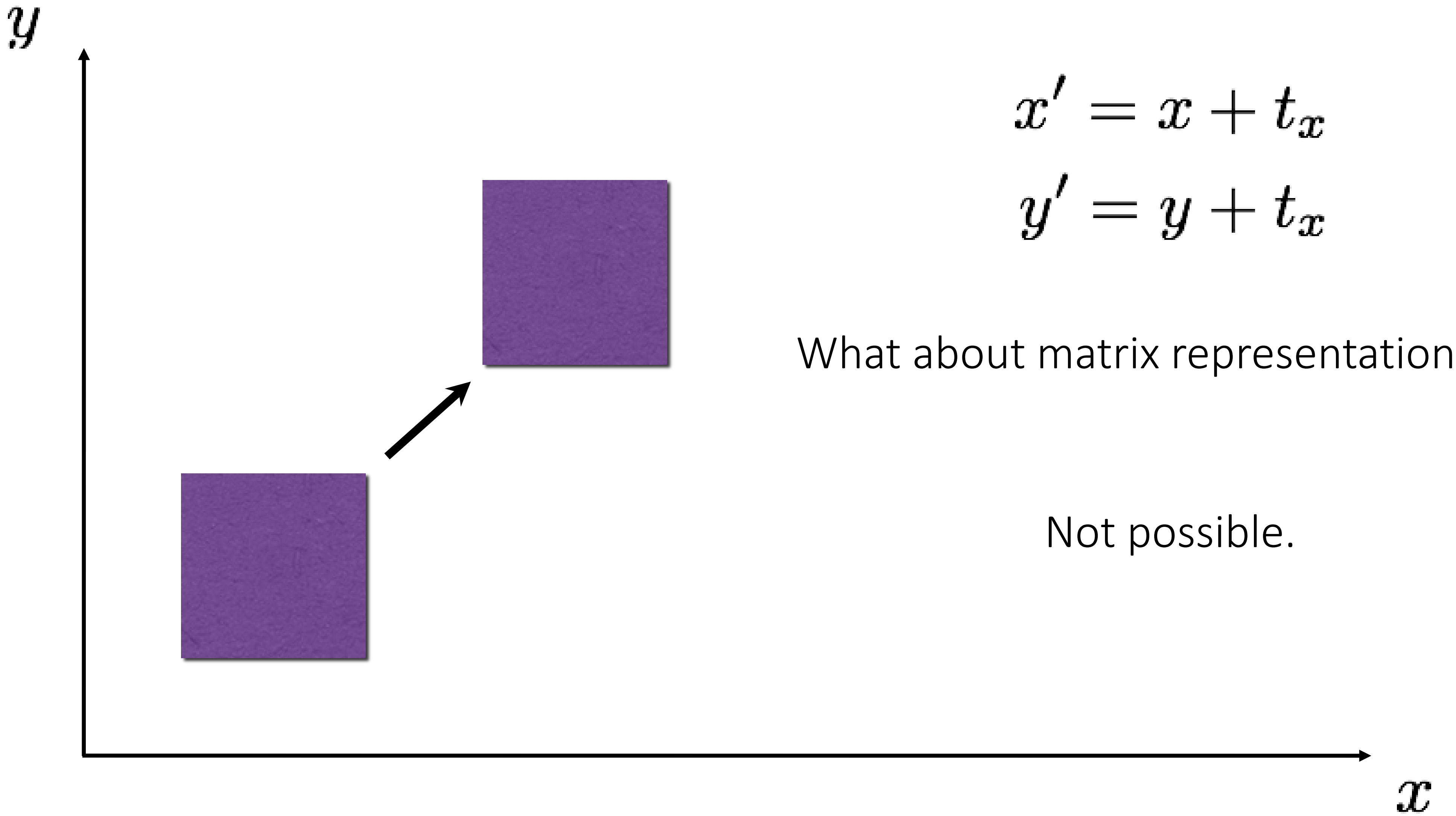


$$x' = x + t_x$$
$$y' = y + t_x$$

What about matrix representation?

$$\mathbf{M} = \left[\begin{array}{cc} ? & ? \\ ? & ? \end{array} \right]$$

2D translation



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

What about matrix representation?

Not possible.

Projective geometry 101

Homogeneous coordinates

heterogeneous coordinates	homogeneous coordinates
$\begin{bmatrix} x \\ y \end{bmatrix}$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

add a 1 here

- Represent 2D point with a 3D vector

Homogeneous coordinates

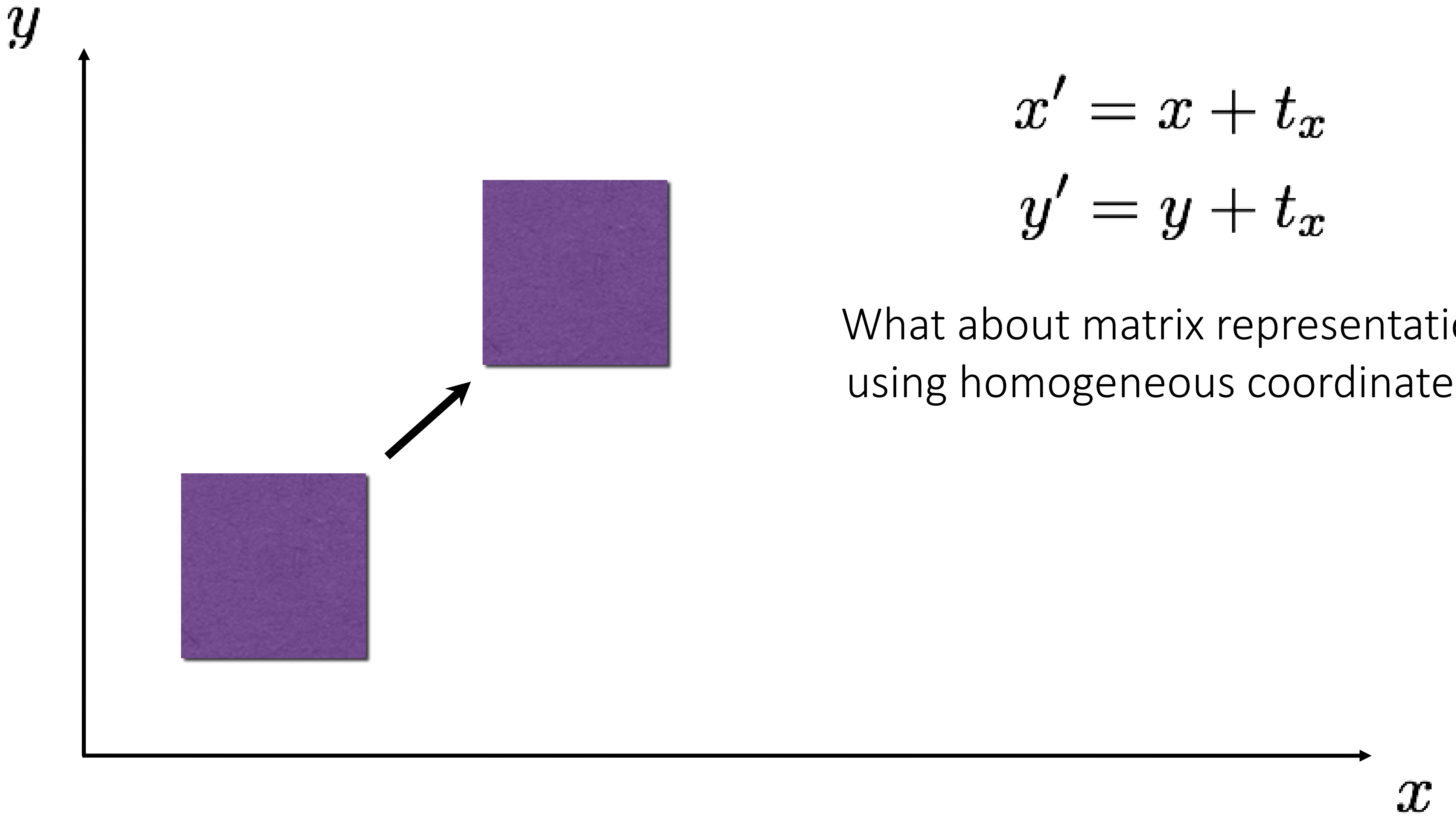
heterogeneous coordinates

homogeneous coordinates

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} ax \\ ay \\ a \end{bmatrix}$$

- Represent 2D point with a 3D vector
- 3D vectors are only defined up to scale

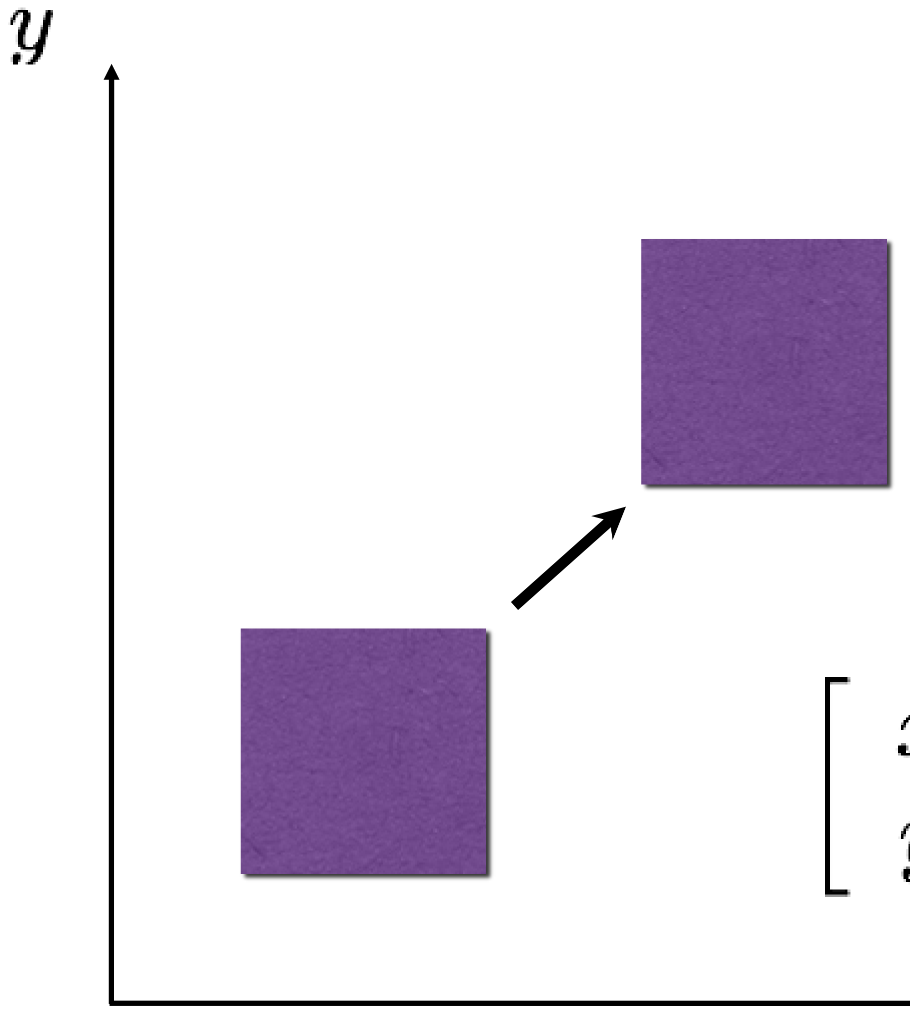
2D translation



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

What about matrix representation using homogeneous coordinates?

2D translation



$$x' = x + t_x$$

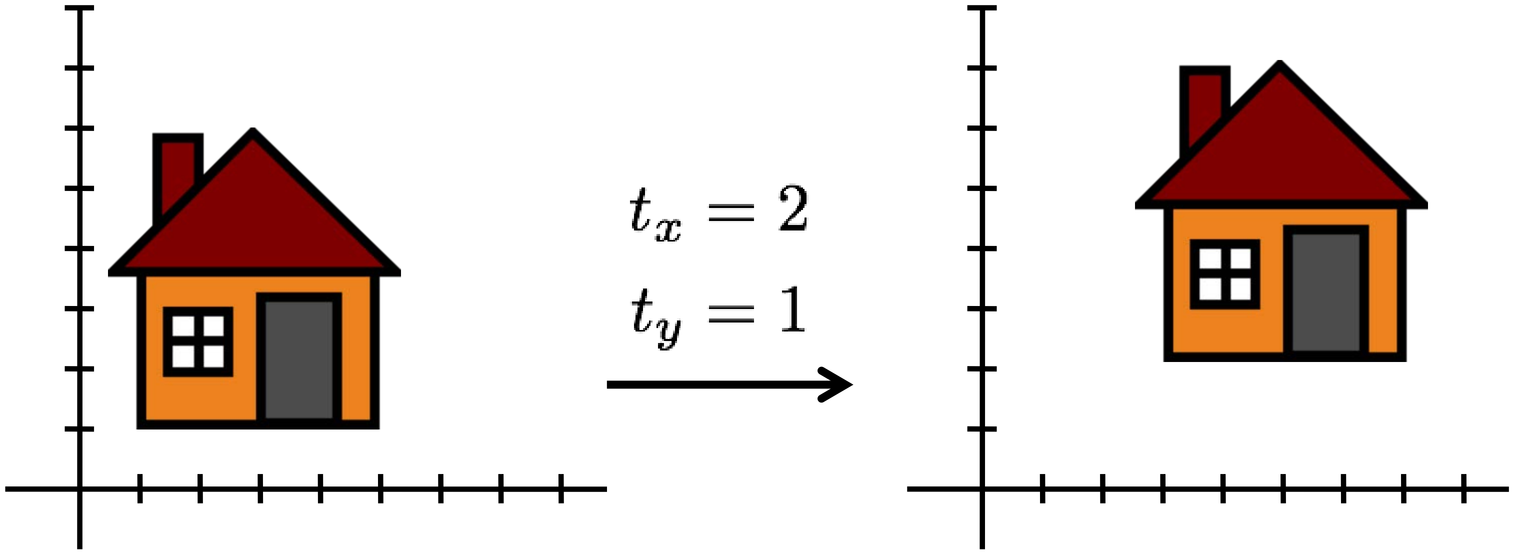
$$y' = y + t_x$$

What about matrix representation using heterogeneous coordinates?

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{M} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

2D translation using homogeneous coordinates

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



Homogeneous coordinates

Conversion:

- heterogeneous → homogeneous

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- homogeneous → heterogeneous

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

- scale invariance

$$\begin{bmatrix} x & y & w \end{bmatrix}^\top = \lambda \begin{bmatrix} x & y & w \end{bmatrix}^\top$$

Special points:

- point at infinity

$$\begin{bmatrix} x & y & 0 \end{bmatrix}$$

- undefined

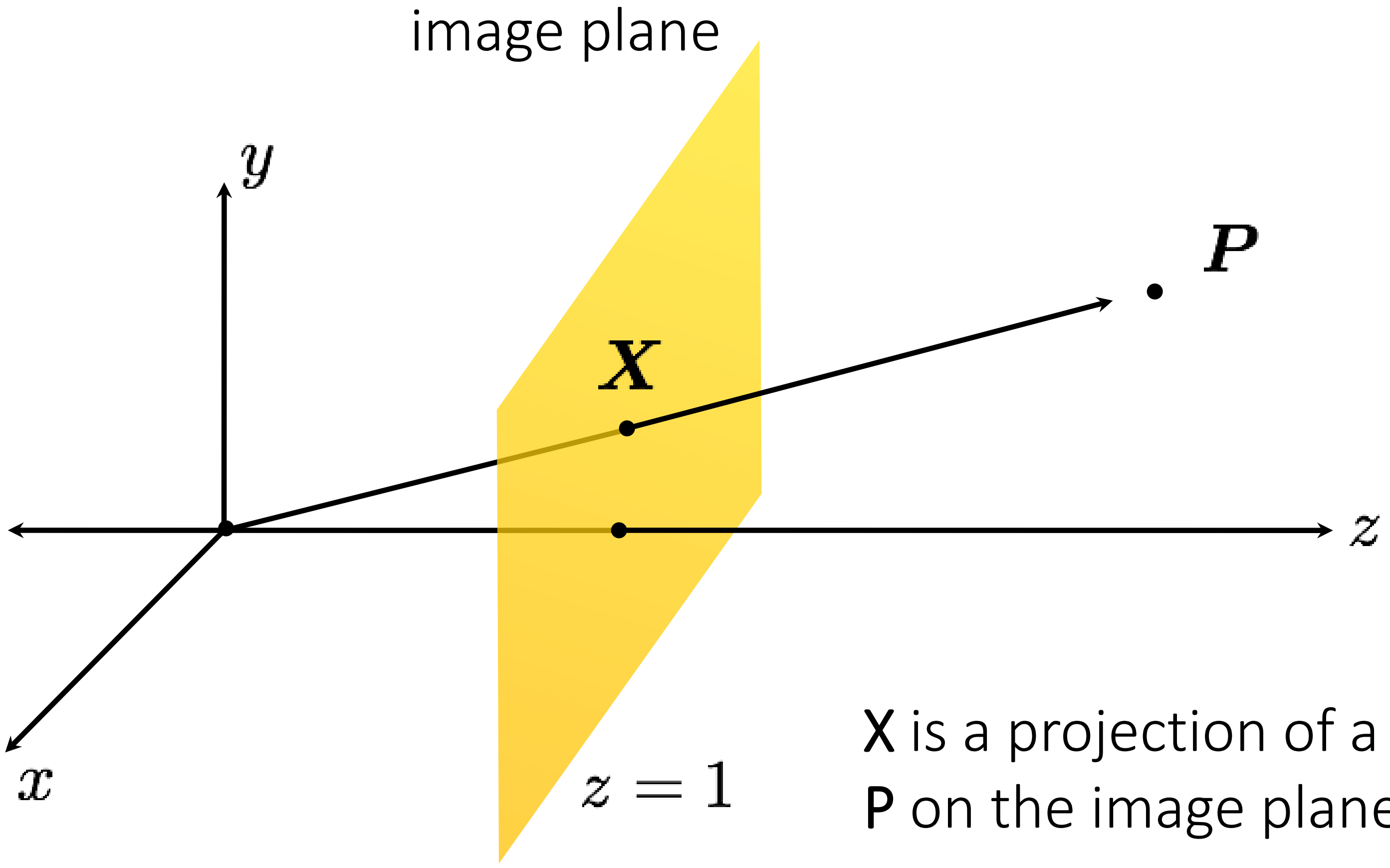
$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Projective geometry

image point in pixel coordinates $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$\Updownarrow$

image point in homogeneous coordinates $\boldsymbol{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$



X is a projection of a point P on the image plane

What does scaling X correspond to?

Transformations in projective geometry

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

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

scaling

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

rotation

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

shearing

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

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

scaling

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

rotation

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

shearing

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

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

scaling

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

rotation

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

shearing

2D transformations in heterogeneous coordinates

Re-write these transformations as 3x3 matrices:

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

translation

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

scaling

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

rotation

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

shearing

Matrix composition

Transformations can be combined by matrix multiplication:

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

p' = ? ? ? **p**

Matrix composition

Transformations can be combined by matrix multiplication:

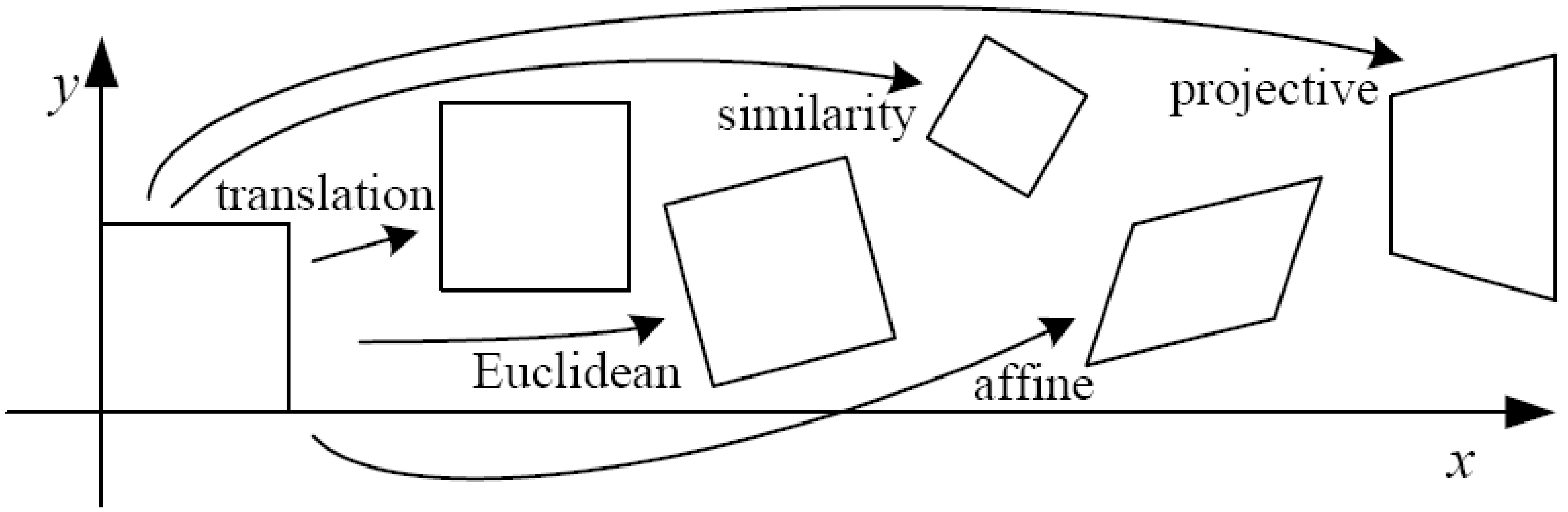
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)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

p' = translation(t_x,t_y) rotation(θ) scale(s,s) **p**

Does the multiplication order matter?

Classification of 2D transformations

Classification of 2D transformations



Classification of 2D transformations

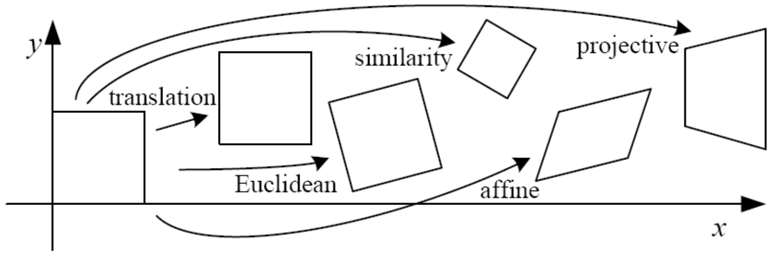
Name	Matrix	# D.O.F.
translation	$\left[\, \boldsymbol{I} \mid \boldsymbol{t} \,\right]$	?
rigid (Euclidean)	$\left[\, \boldsymbol{R} \mid \boldsymbol{t} \,\right]$	?
similarity	$\left[\, s\boldsymbol{R} \mid \boldsymbol{t} \,\right]$	?
affine	$\left[\, \boldsymbol{A} \,\right]$	?
projective	$\left[\, \tilde{\boldsymbol{H}} \,\right]$	?

Classification of 2D transformations

Translation:

$$\begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom?

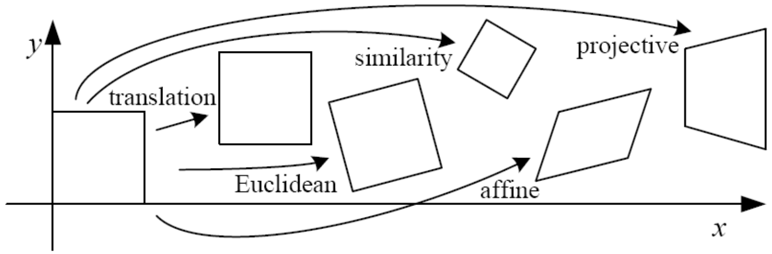


Classification of 2D transformations

Euclidean (rigid):
rotation + translation

$$\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?

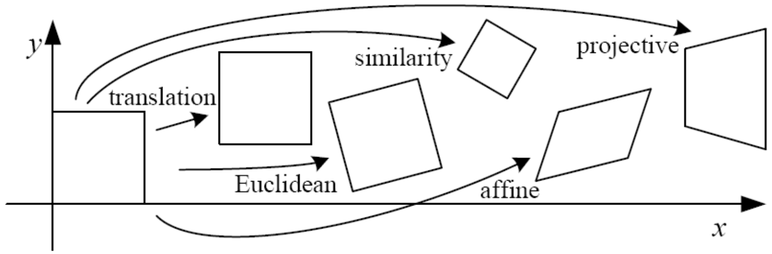


Classification of 2D transformations

Euclidean (rigid):
rotation + translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & r_3 \\ \sin\theta & \cos\theta & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom?

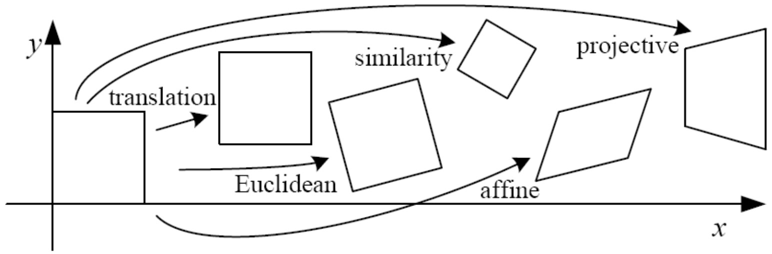


Classification of 2D transformations

Similarity:
uniform scaling + rotation
+ translation

$$\begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?



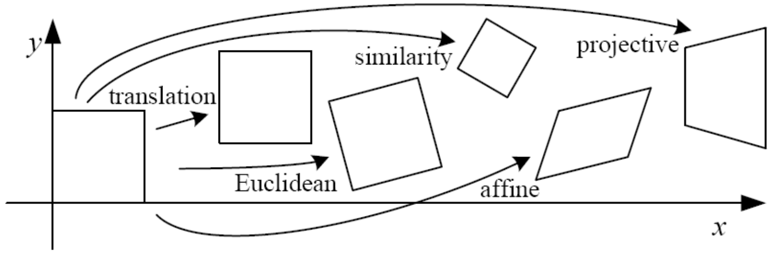
Classification of 2D transformations

multiply these four by scale **s**

Similarity:
uniform scaling + rotation
+ translation

$$\begin{bmatrix} \boxed{\begin{matrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{matrix}} & \begin{matrix} r_3 \\ r_6 \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

How many degrees of freedom?

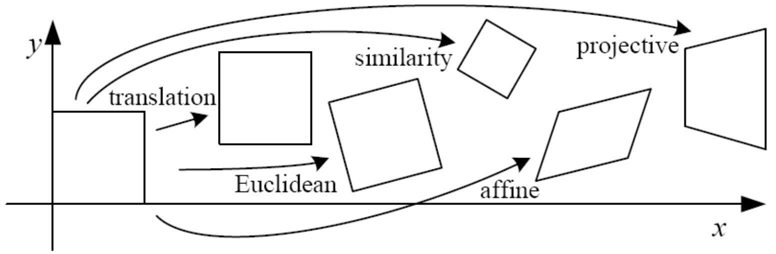


Classification of 2D transformations

Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?



Classification of 2D transformations

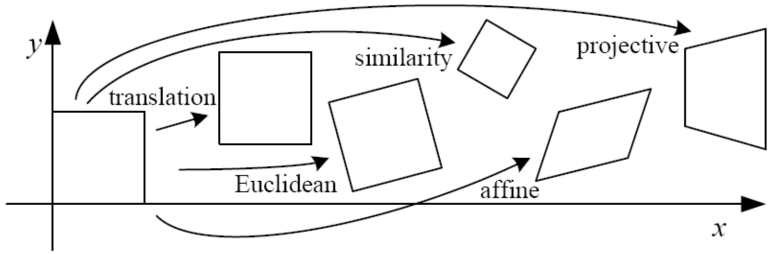
Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Are there any values that are related?

similarity shear

$$\begin{bmatrix} sr_1 & sr_2 \\ sr_3 & sr_4 \end{bmatrix} \begin{bmatrix} 1 & h_1 \\ h_2 & 1 \end{bmatrix} = \begin{bmatrix} sr_1 + h_2 sr_2 & sr_2 + h_1 sr_1 \\ sr_3 + h_2 sr_4 & sr_4 + h_1 sr_3 \end{bmatrix}$$



Classification of 2D transformations

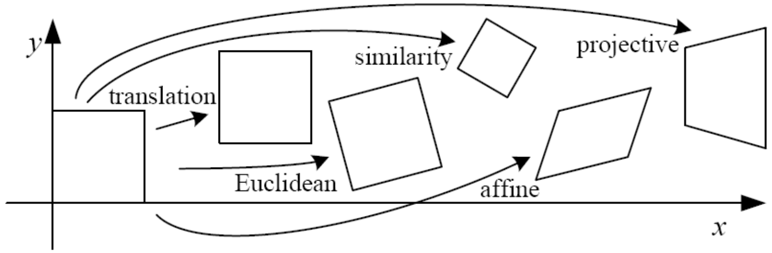
Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom?

similarity shear

$$\begin{bmatrix} sr_1 & sr_2 \\ sr_3 & sr_4 \end{bmatrix} \begin{bmatrix} 1 & h_1 \\ h_2 & 1 \end{bmatrix} = \begin{bmatrix} sr_1 + h_2 sr_2 & sr_2 + h_1 sr_1 \\ sr_3 + h_2 sr_4 & sr_4 + h_1 sr_3 \end{bmatrix}$$



Affine transformations

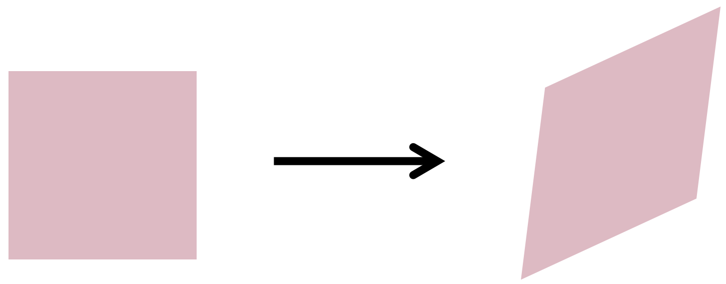
Affine transformations are combinations of

- arbitrary (4-DOF) linear transformations; and
- translations

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

Properties of affine transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines map to parallel lines
- ratios are preserved
- compositions of affine transforms are also affine transforms



Does the last coordinate w ever change?

Affine transformations

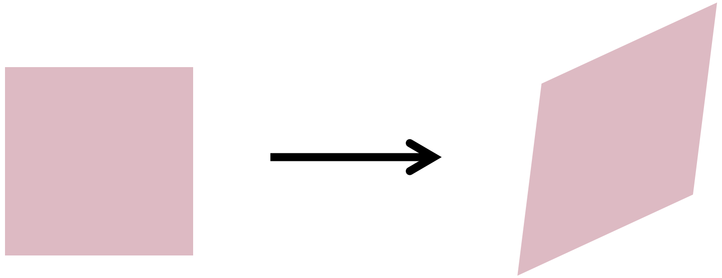
Affine transformations are combinations of

- arbitrary (4-DOF) linear transformations; and
- translations

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

Properties of affine transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines map to parallel lines
- ratios are preserved
- compositions of affine transforms are also affine transforms



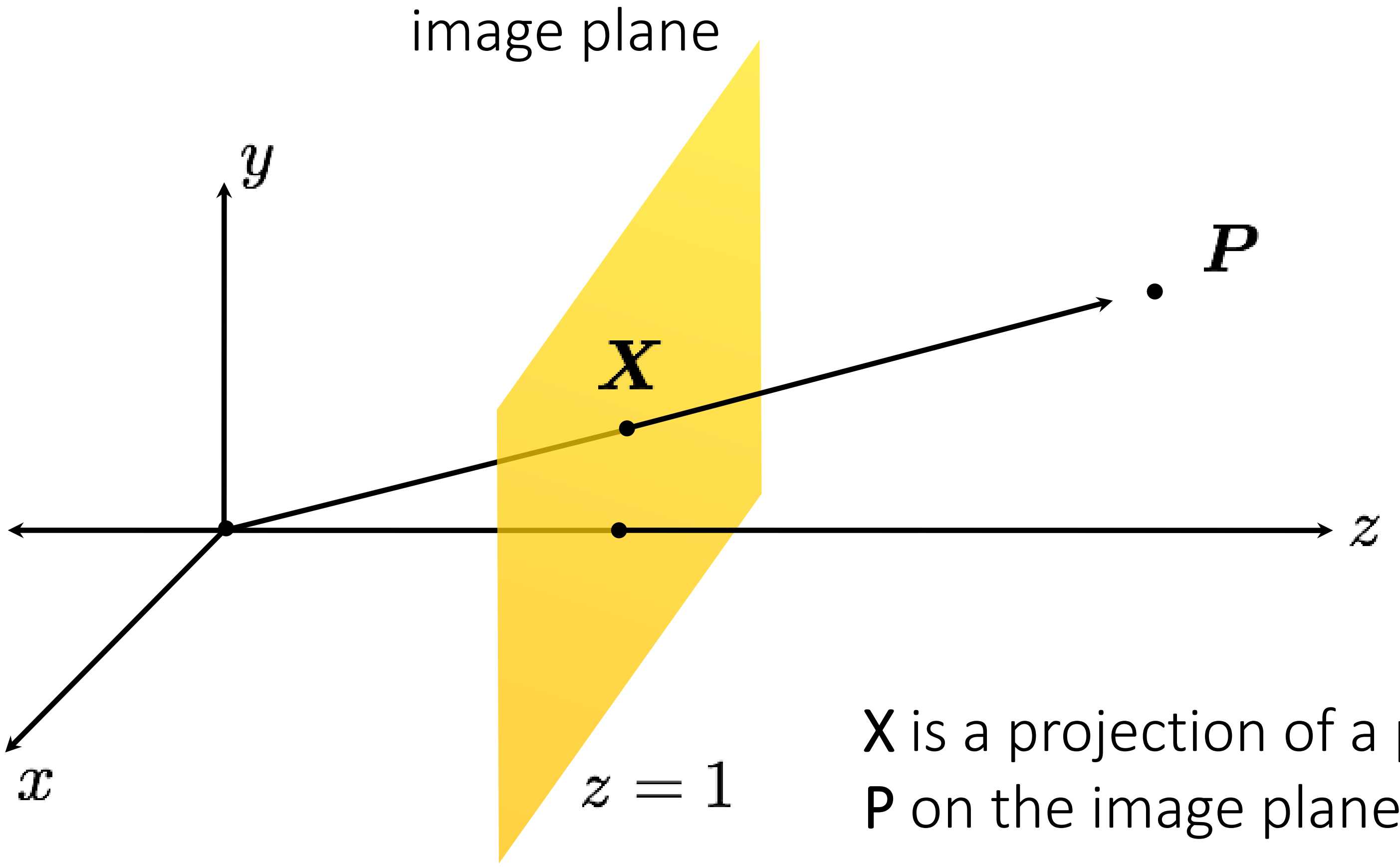
Nope! But what does that mean?

How to interpret affine transformations here?

image point in pixel coordinates $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$\Updownarrow$

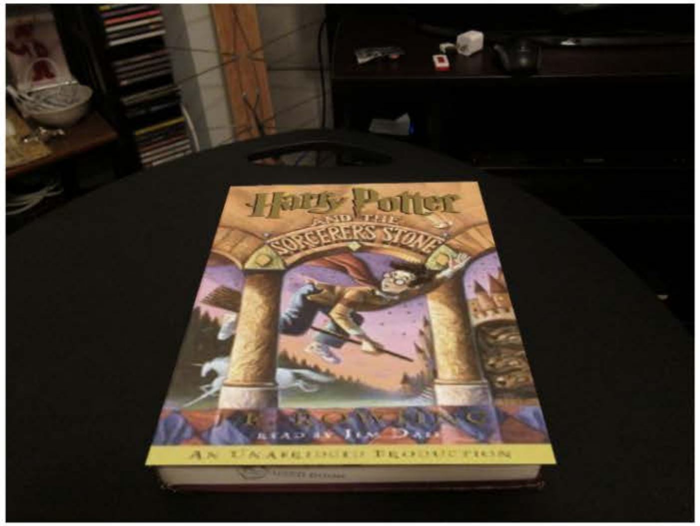
image point in heterogeneous coordinates $\boldsymbol{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

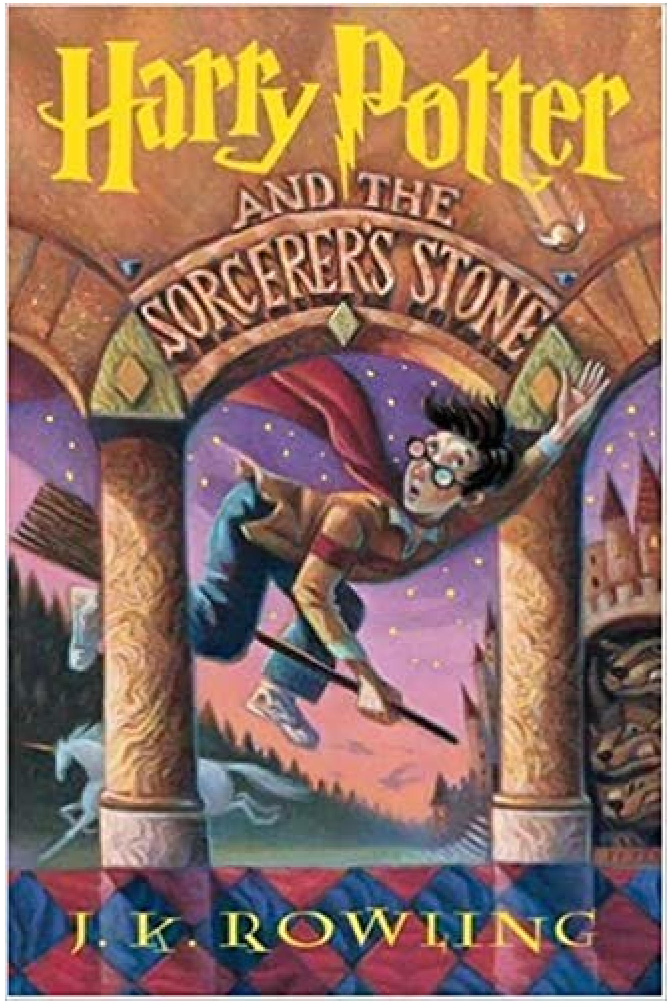


X is a projection of a point P on the image plane

Move point on yellow plane only *inside* the plane

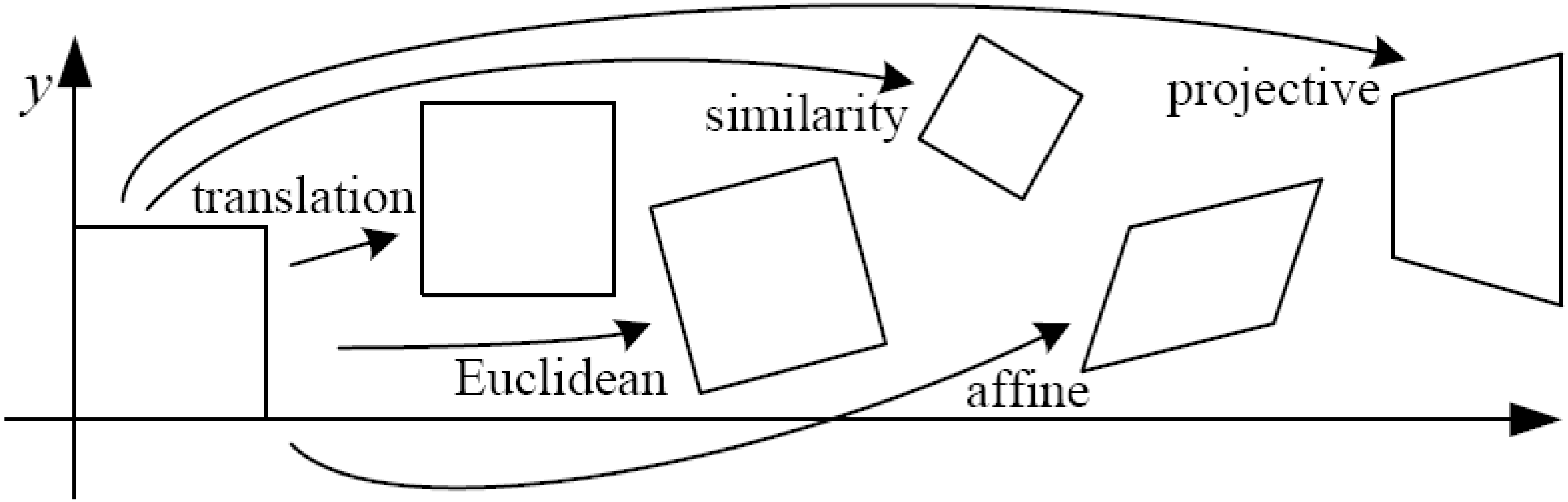
This is not an affine transformation





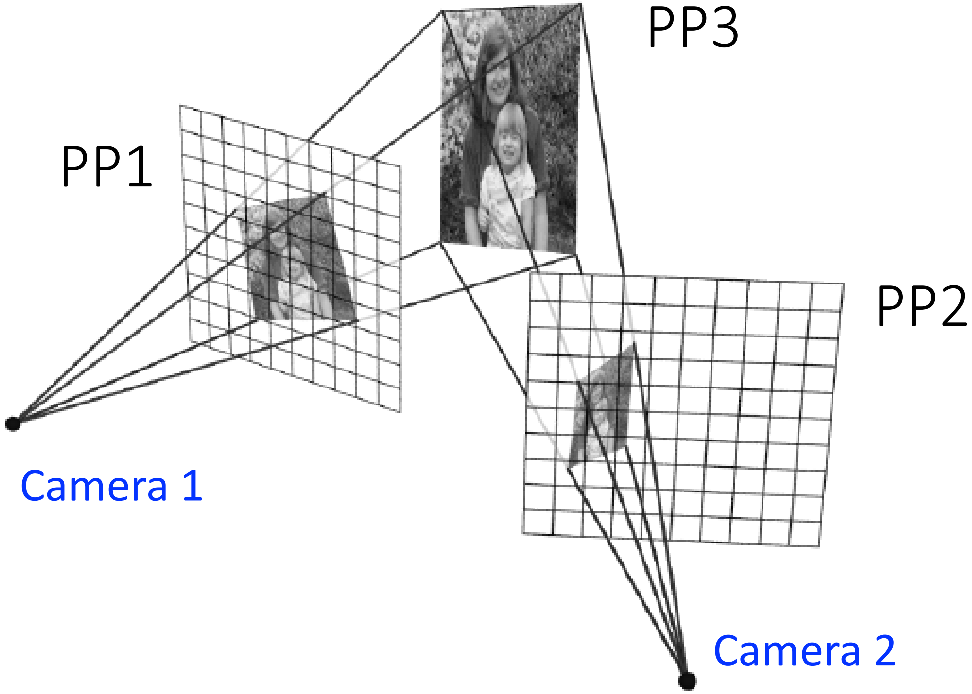
e.g. parallel lines do not map to parallel lines

Classification of 2D transformations



Which kind transformation is needed to warp projective plane 1 into projective plane 2?

- A projective transformation (a.k.a. a homography).



Projective transformations

This is equation up to scale

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

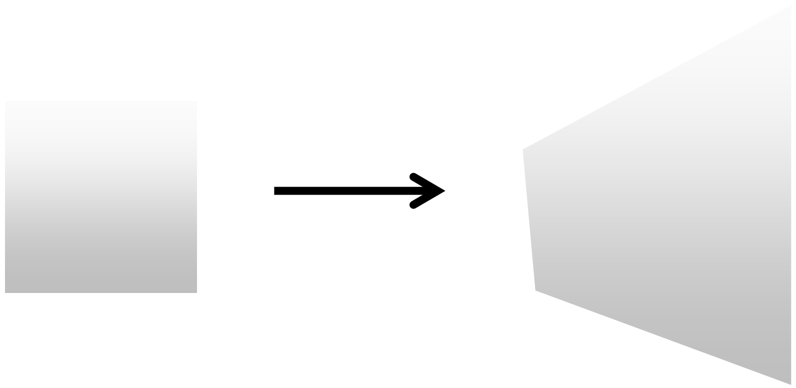
How many degrees of freedom?

Projective transformations are combinations of

- affine transformations; and
- projective wraps

Properties of projective transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms



Projective transformations

This is equation up to scale

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

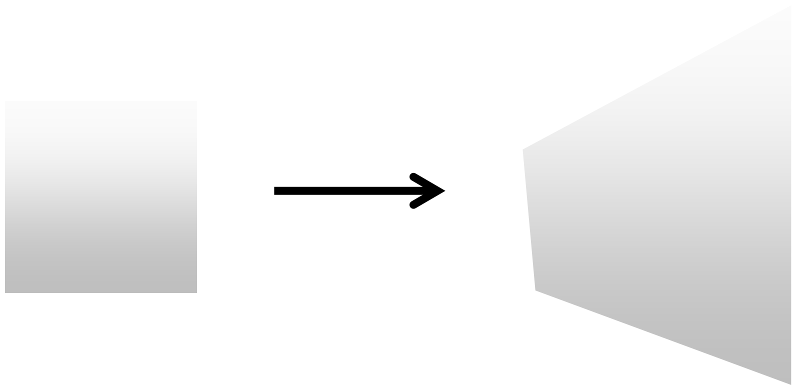
8 DOF: vectors (and therefore matrices) are defined up to scale)

Projective transformations are combinations of

- affine transformations; and
- projective wraps

Properties of projective transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms

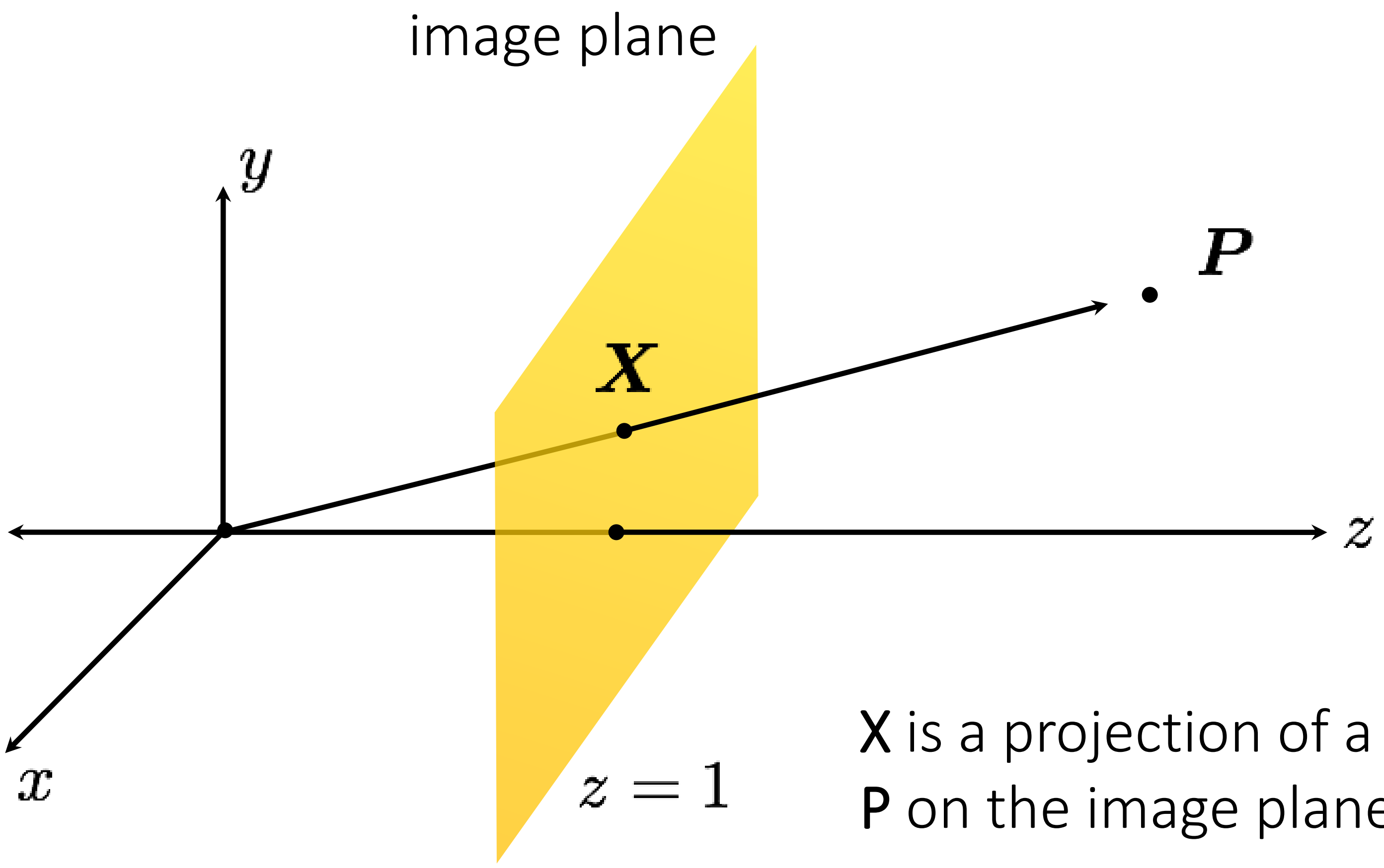


How to interpret projective transformations here?

image point in pixel coordinates $\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$\Updownarrow$$

image point in heterogeneous coordinates $\boldsymbol{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

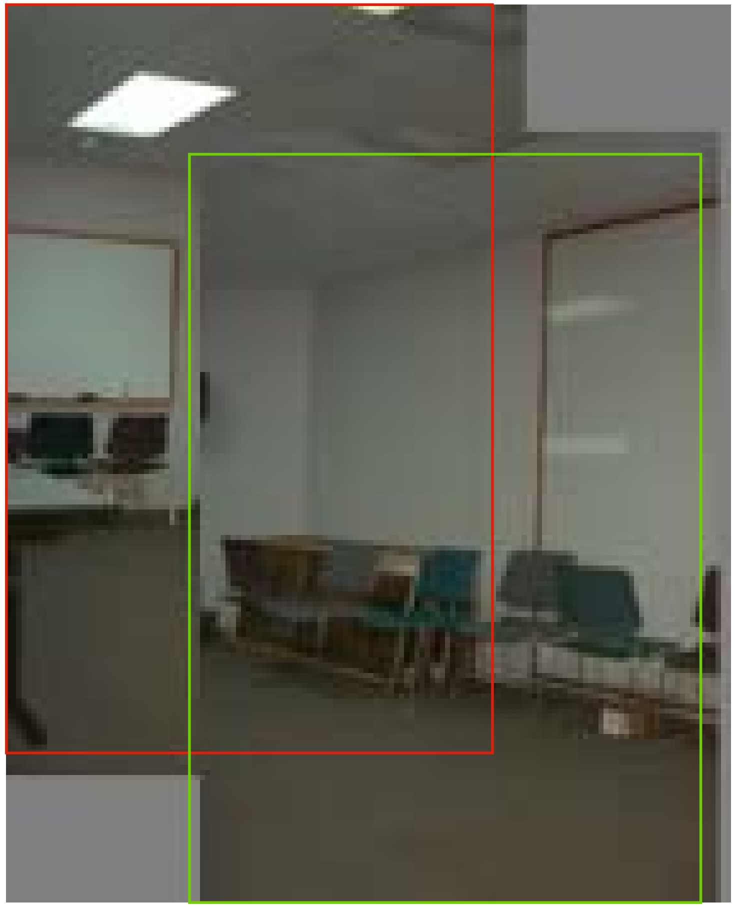


X is a projection of a point P on the image plane

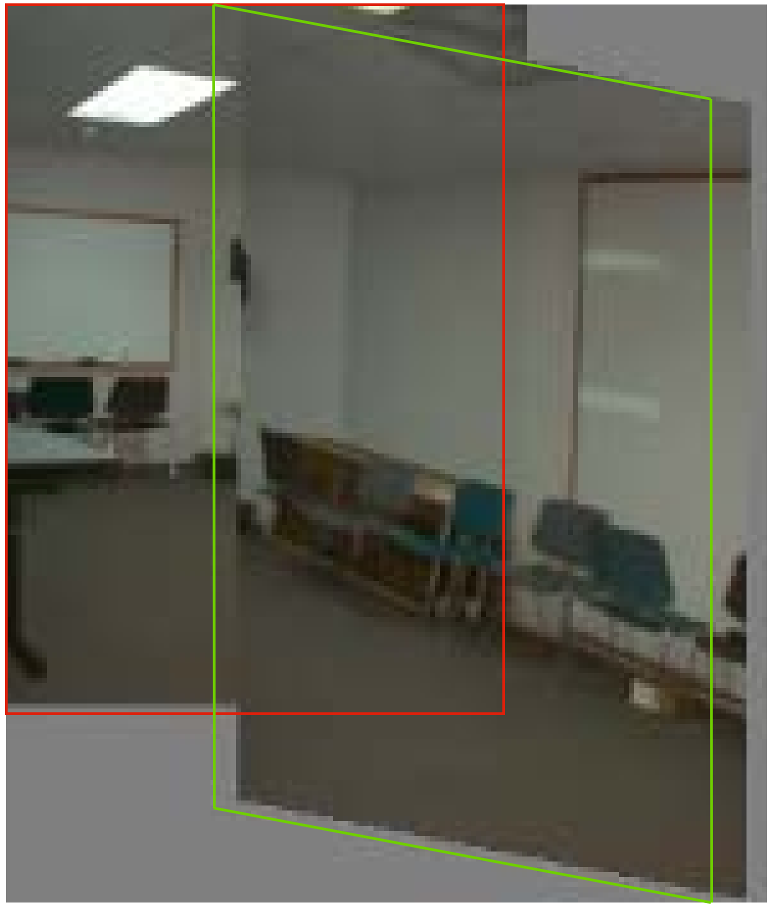
Move point on yellow plane *outside* the plane

Warping with different transformations

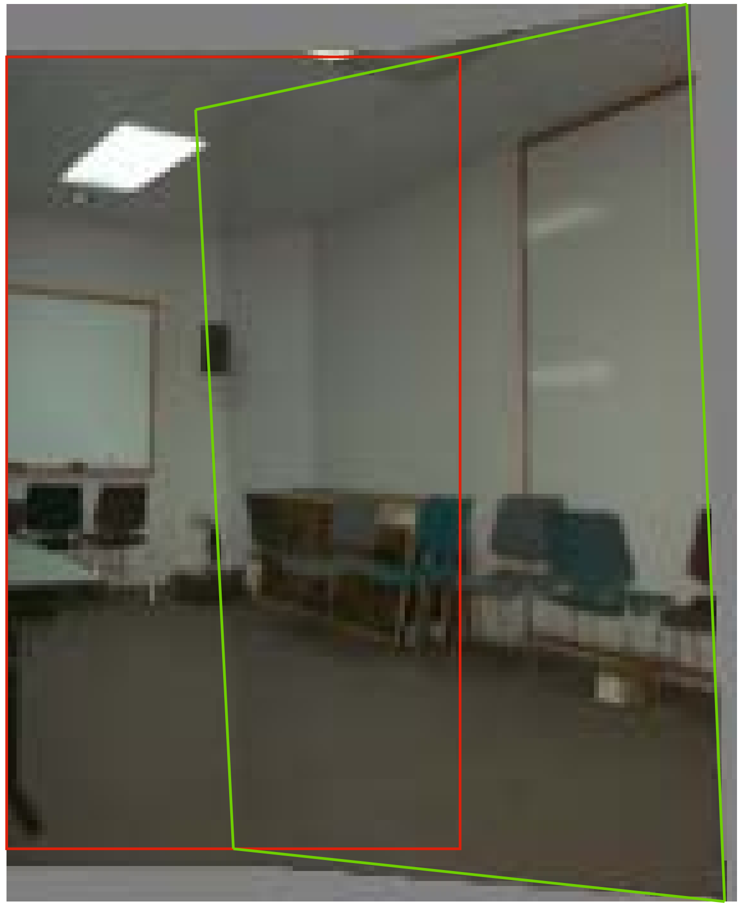
translation



affine



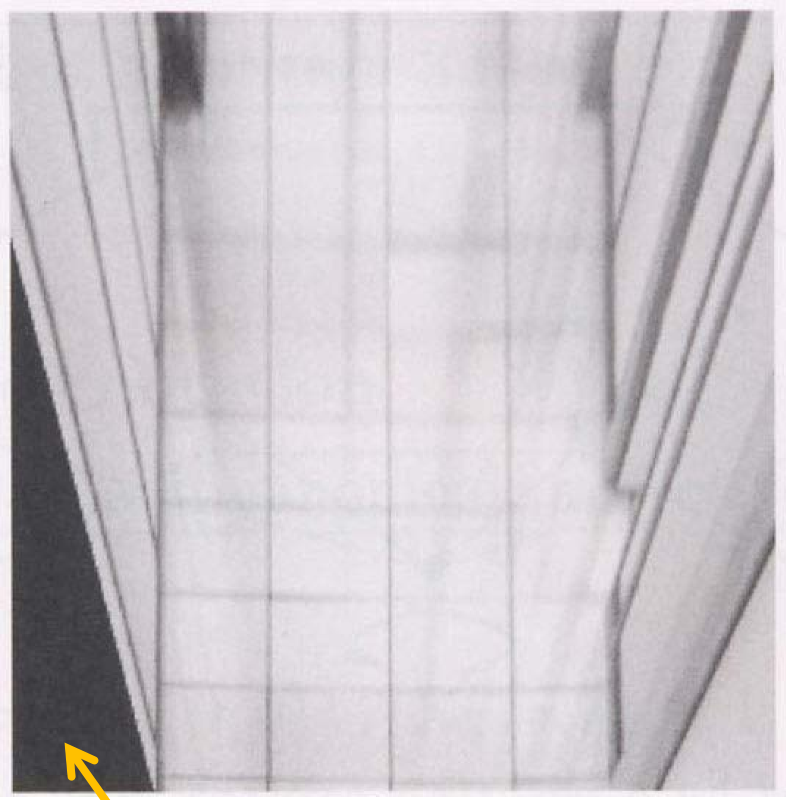
pProjective (homography)



View warping

original view

synthetic top view



synthetic side view

What are these black areas near the boundaries?

Virtual camera rotations

original view

synthetic rotations

Image rectification

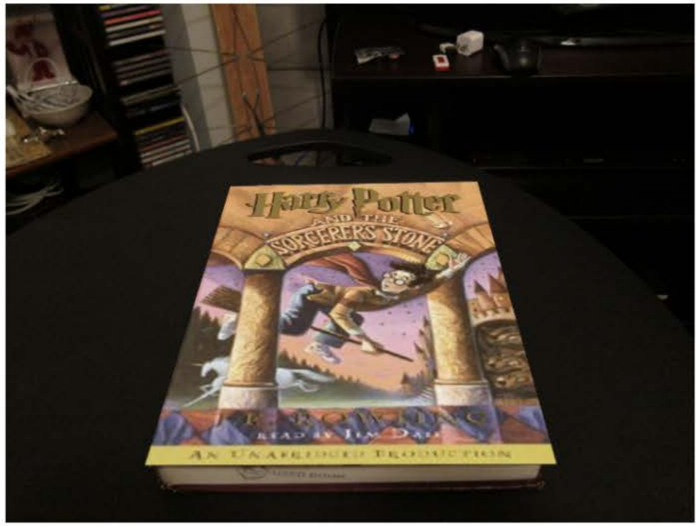
two
original
images

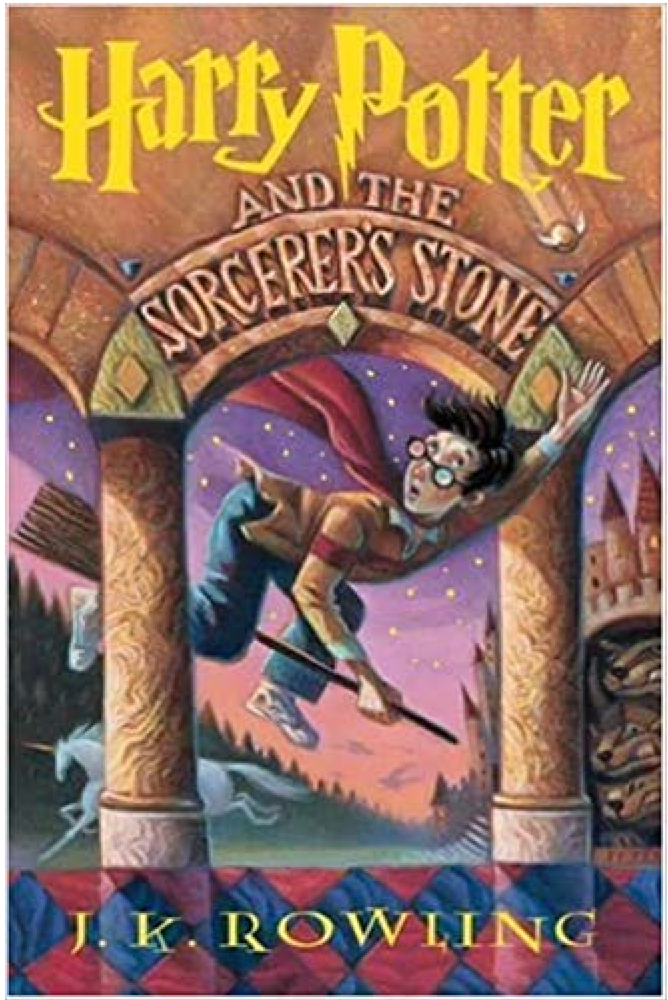
rectified and stitched

Street art



This is not an affine transformation





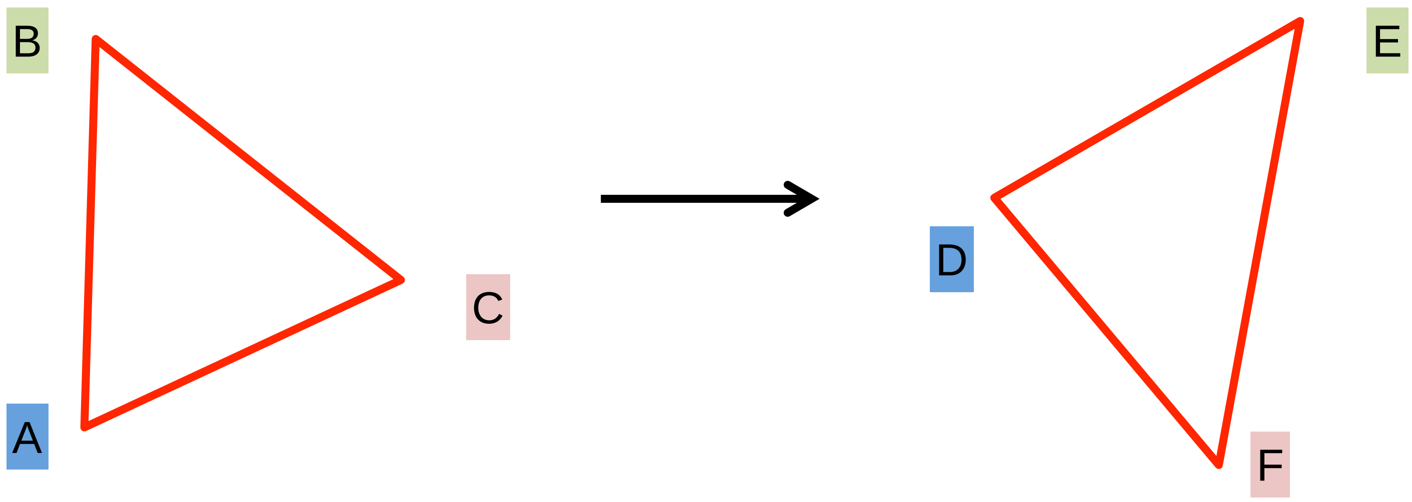
Theorem: Every image of a ***plane*** in camera 1 can be mapped into an image of a plane in camera 2 via a 3x3 projective homograpy

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

Determining unknown 2D transformations

Determining unknown transformations

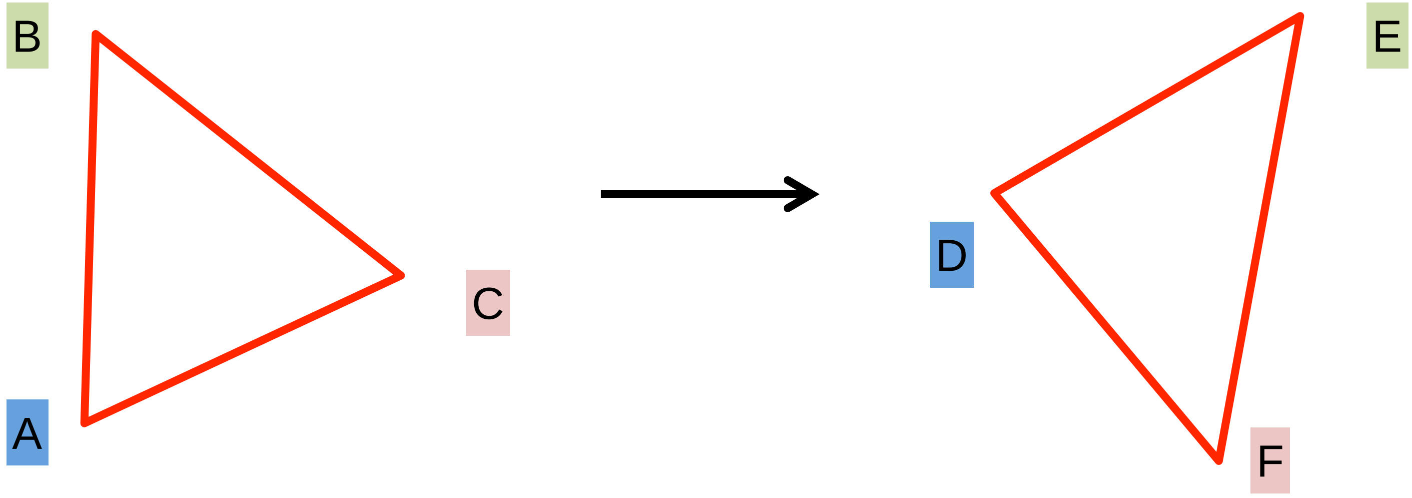
Suppose we have two triangles: ABC and DEF.

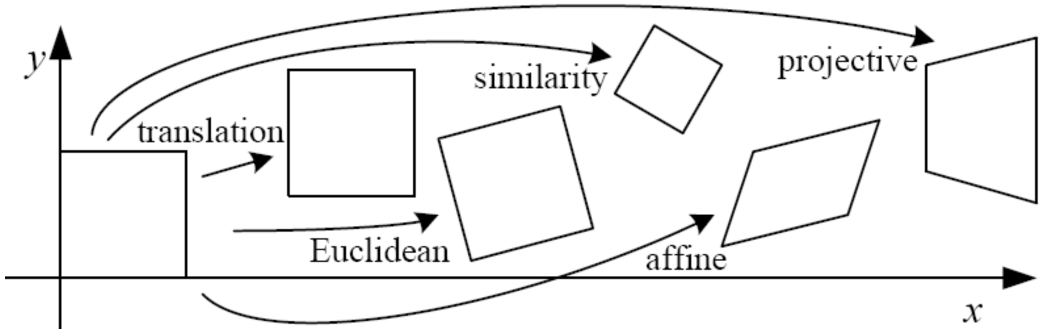


Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?

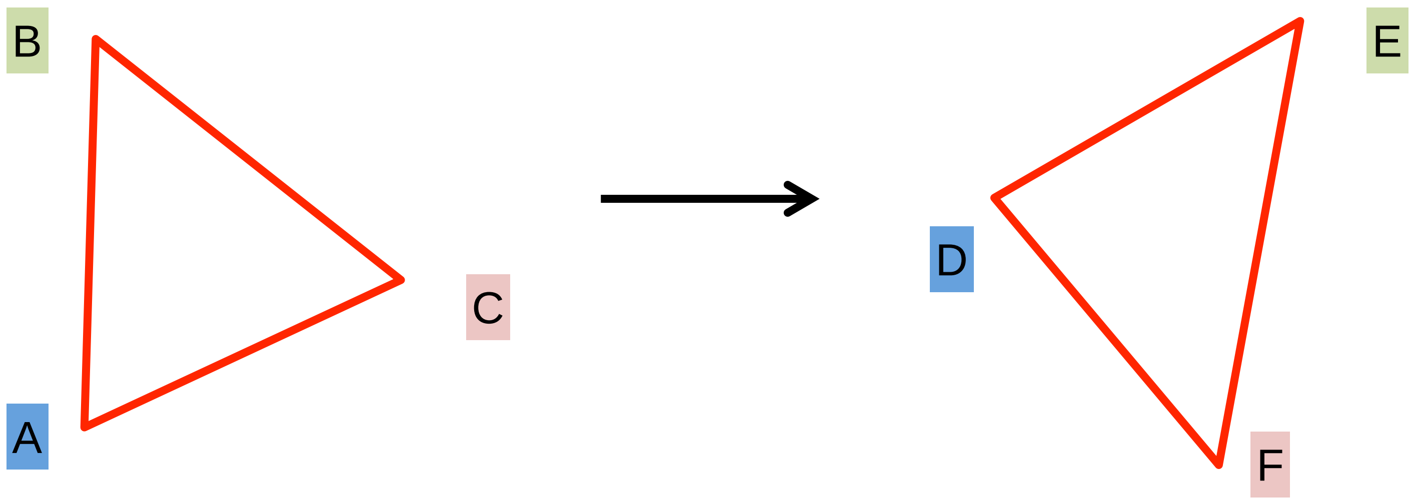




Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
- How do we determine the unknown parameters?



Affine transform:

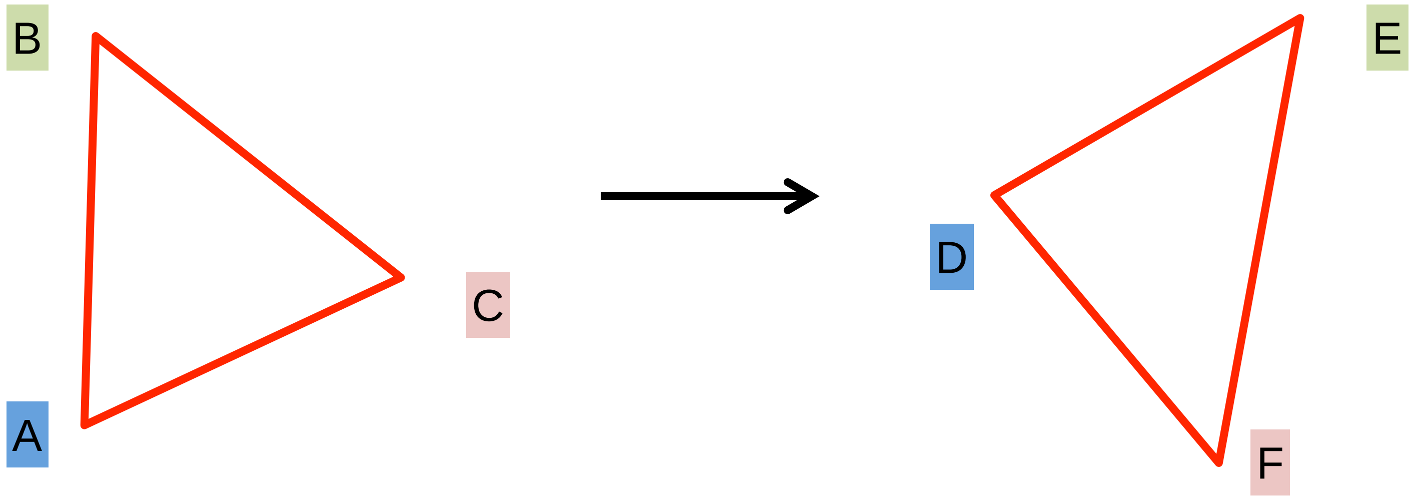
$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom do we have?

Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
- How do we determine the unknown parameters?



unknowns

$$\boldsymbol{x}' = \mathbf{M}\boldsymbol{x}$$

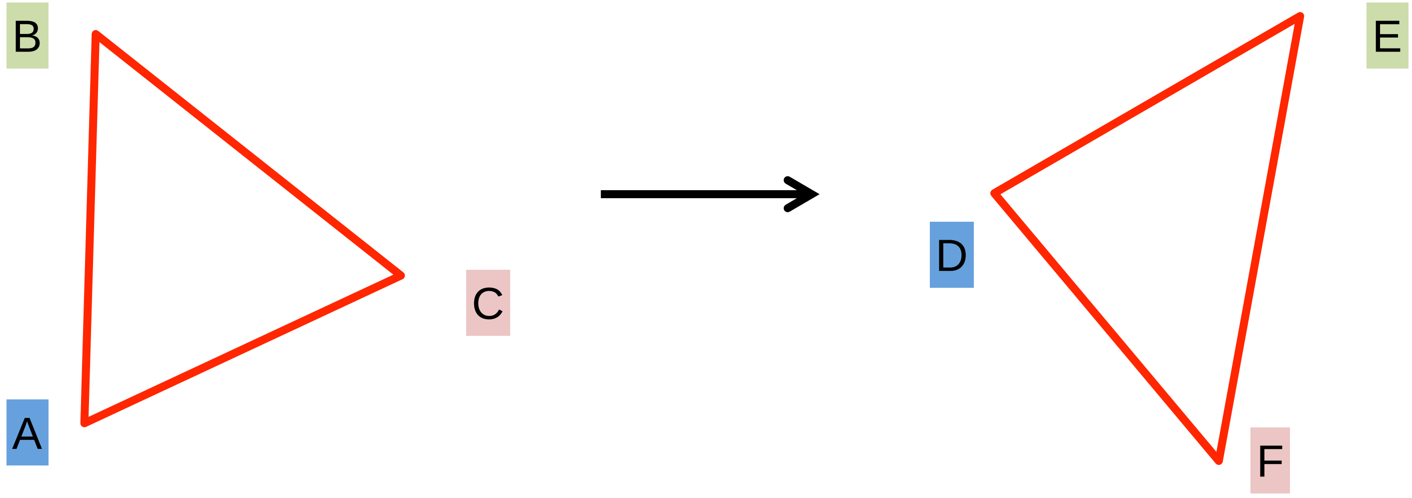
point correspondences

- One point correspondence gives how many equations?
- How many point correspondences do we need?

Determining unknown transformations

Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?
- How do we determine the unknown parameters?

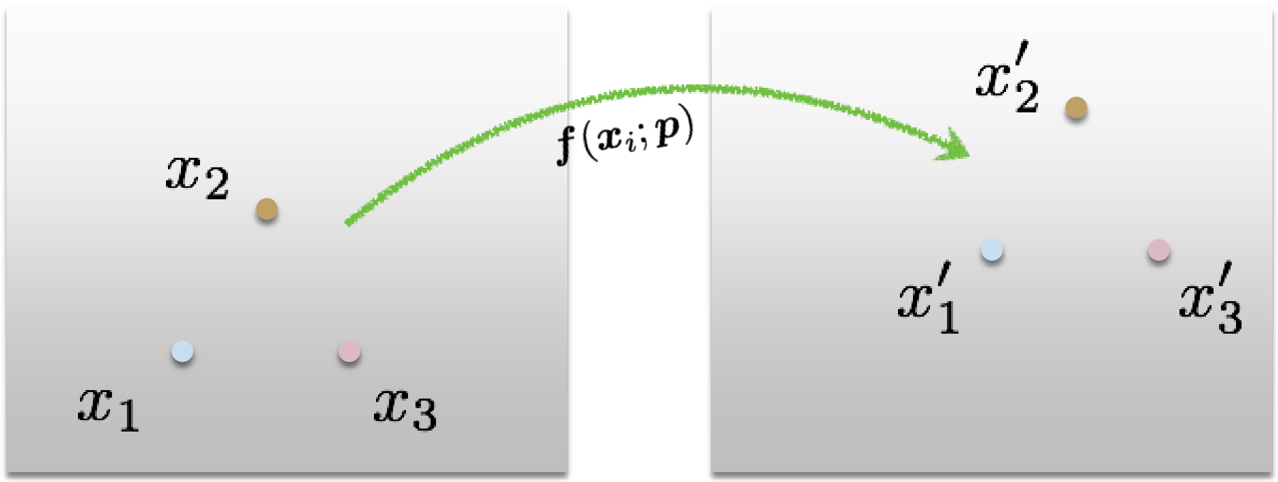


unknowns

$$\boldsymbol{x}' = \mathbf{M}\boldsymbol{x}$$

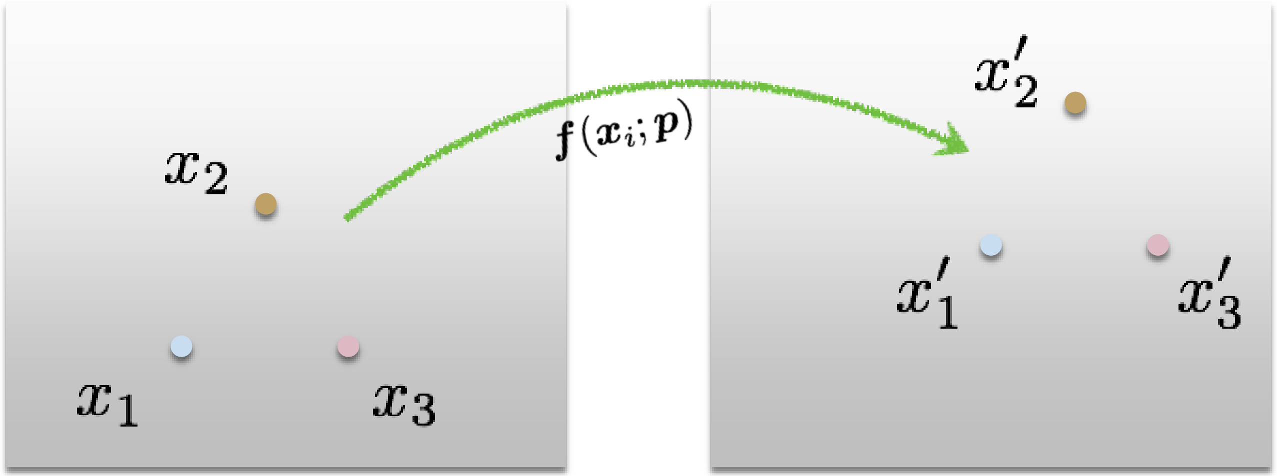
point correspondences

How do we solve this for **M**?



Least Squares Error

$$E_{\mathrm{LS}} = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i\|^2$$



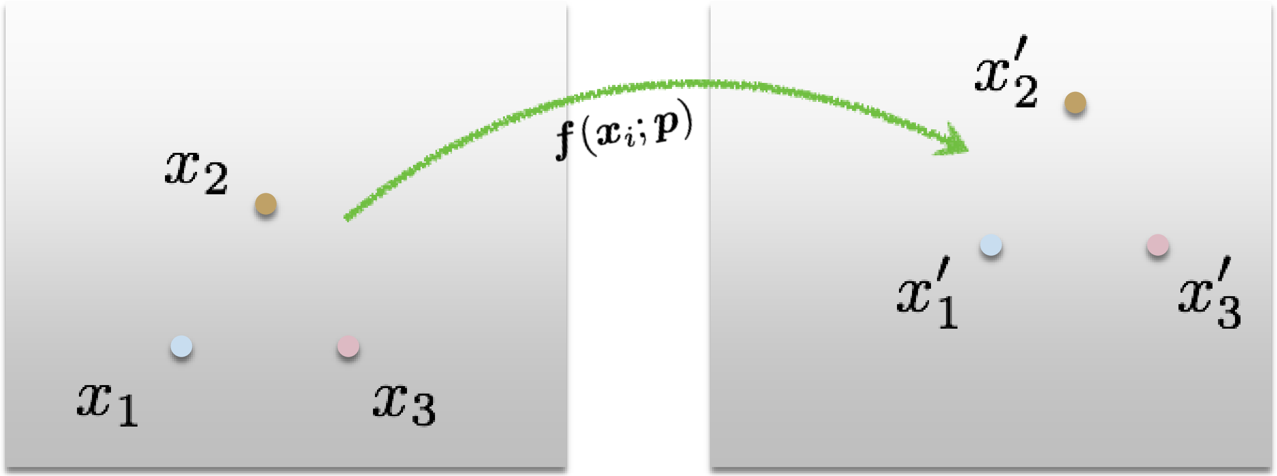
Least Squares Error

What is this?

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

What is this?

What is this?



$$||\boldsymbol{x}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

Least Squares Error

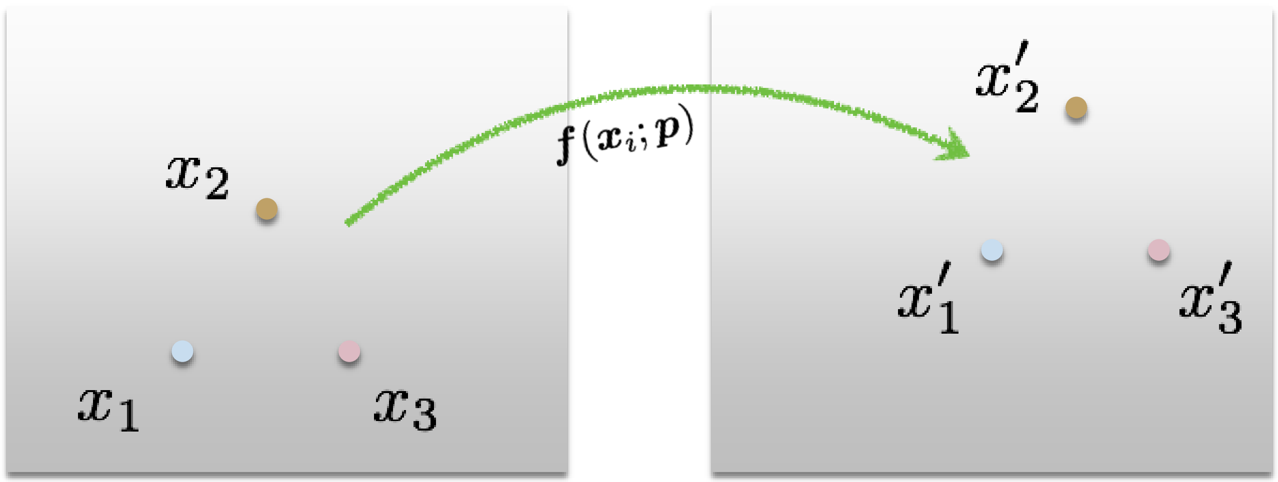
Euclidean (L2) norm

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

squared!

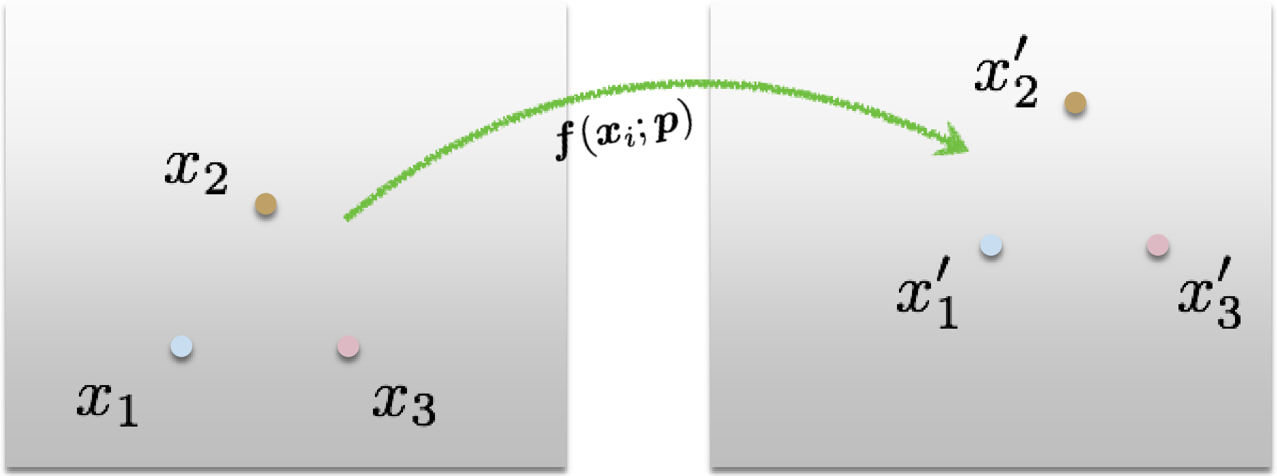
predicted location

measured location



Least Squares Error

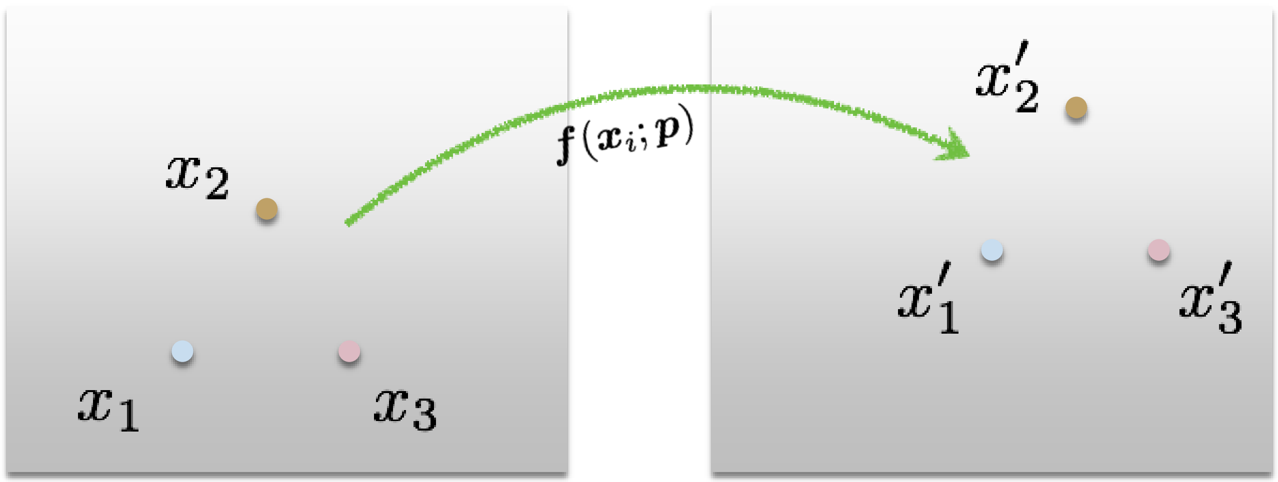
$$E_{\mathrm{LS}} = \sum_i \underbrace{\| \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i \|}_{\text{Residual (projection error)}}{}^2$$



Least Squares Error

$$E_{\mathrm{LS}} = \sum_i \| \boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i \|^2$$

What is the free variable?
What do we want to optimize?



Find parameters that minimize squared error

$$\hat{\boldsymbol{p}} = \arg\min_{\boldsymbol{p}} \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i\|^2$$

General form of linear least squares

(**Warning:** change of notation. x is a vector of parameters!)

$$E_{\mathrm{LLS}} = \sum_i |\boldsymbol{a}_i \boldsymbol{x} - \boldsymbol{b}_i|^2$$

$$= \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 \qquad \text{(matrix form)}$$

Determining unknown transformations

Affine transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Why can we drop the last line?

Vectorize transformation parameters:

Stack equations from point correspondences:

Corresponding point pair 1

Corresponding point pair 2

Corresponding point pair j

$$\underbrace{\begin{bmatrix} x' \\ y' \\ x' \\ y' \\ \vdots \\ x' \\ y' \end{bmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ & & & \vdots & & \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}}_{\boldsymbol{x}}$$

Notation in system form:

General form of linear least squares

(**Warning:** change of notation. x is a vector of parameters!)

$$E_{\mathrm{LLS}} = \sum_i |\boldsymbol{a}_i \boldsymbol{x} - \boldsymbol{b}_i|^2$$

$$= \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 \quad \text{(matrix form)}$$

This function is quadratic.

How do you find the root of a quadratic?

Solving the linear system

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = \boldsymbol{x}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} - 2\boldsymbol{x}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Derivative: $\mathbf{2(A^TA)x - 2A^Tb}$

Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $\boldsymbol{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$ ⟵ Note: You almost <u>never</u> want to compute the inverse of a matrix.

In Matlab:

```
x = A \ b
```

Python:

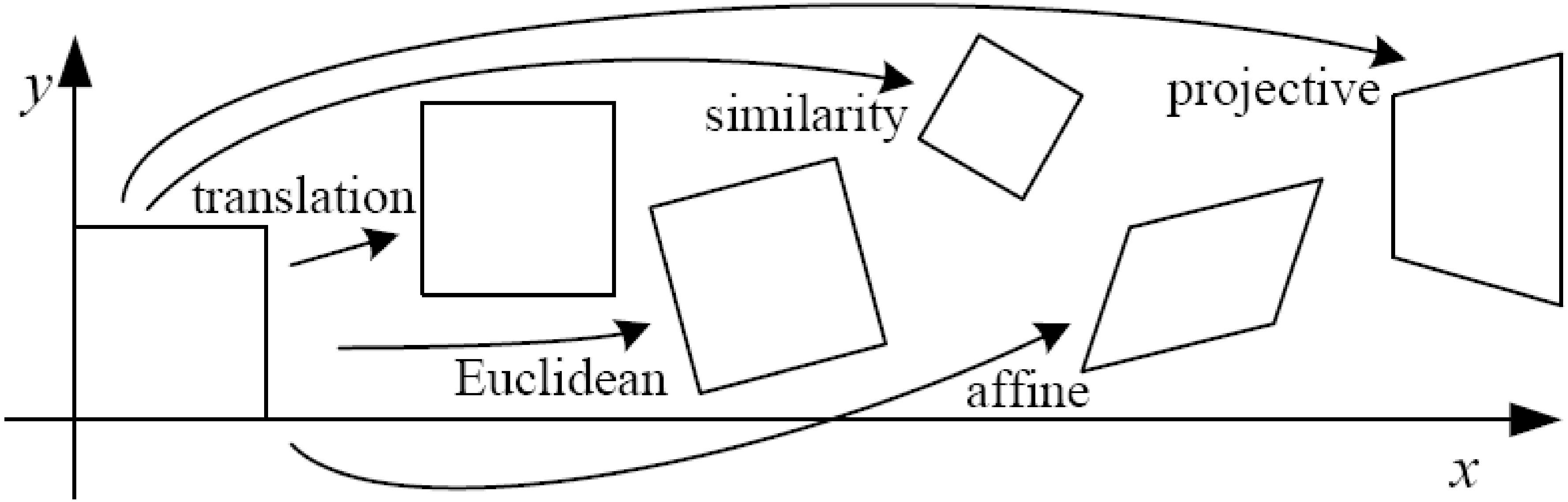
```
Api=scipy.linalg.pinv(A)
```

Linear least squares estimation only works when the transform function is ?

Linear least squares estimation only works when the transform function is **linear! (duh)**

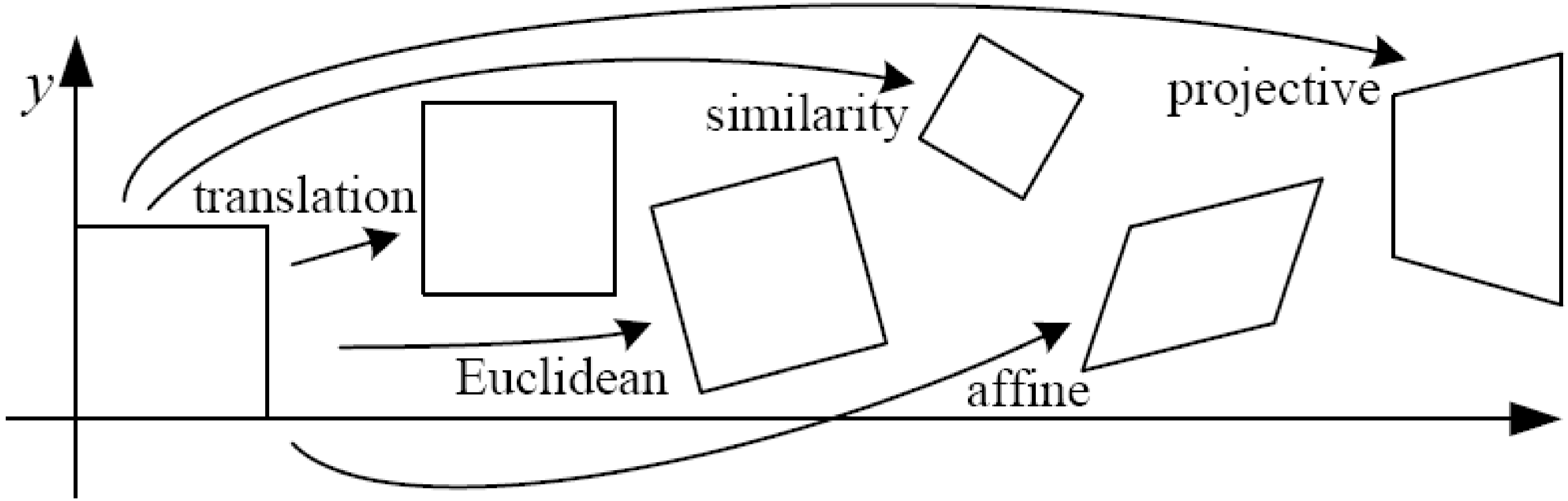
Also doesn’t deal well with outliers

Classification of 2D transformations

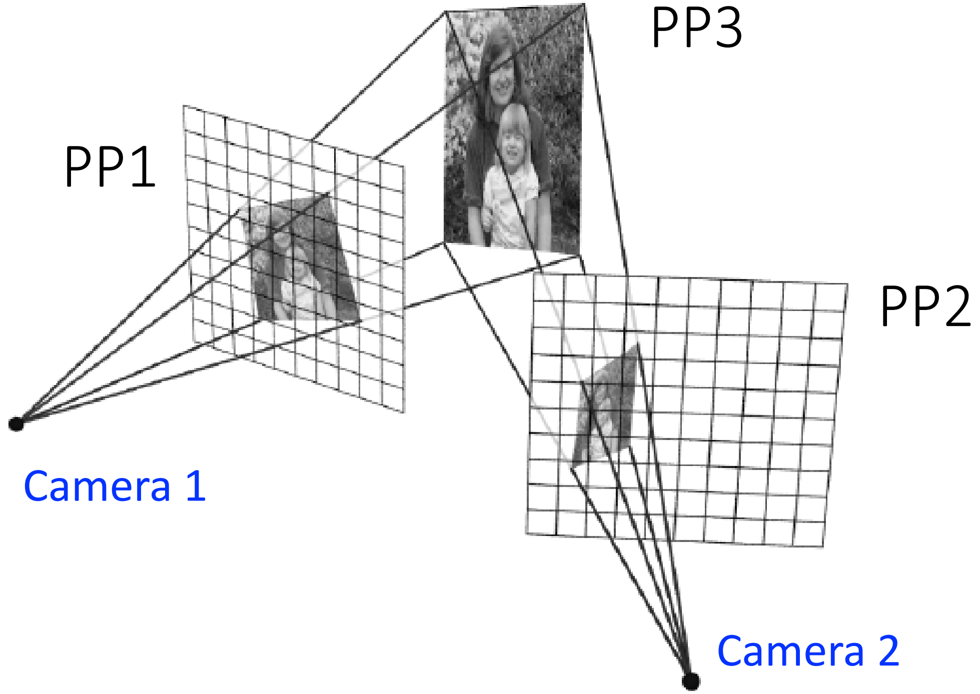


Name	Matrix	# D.O.F.
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	8

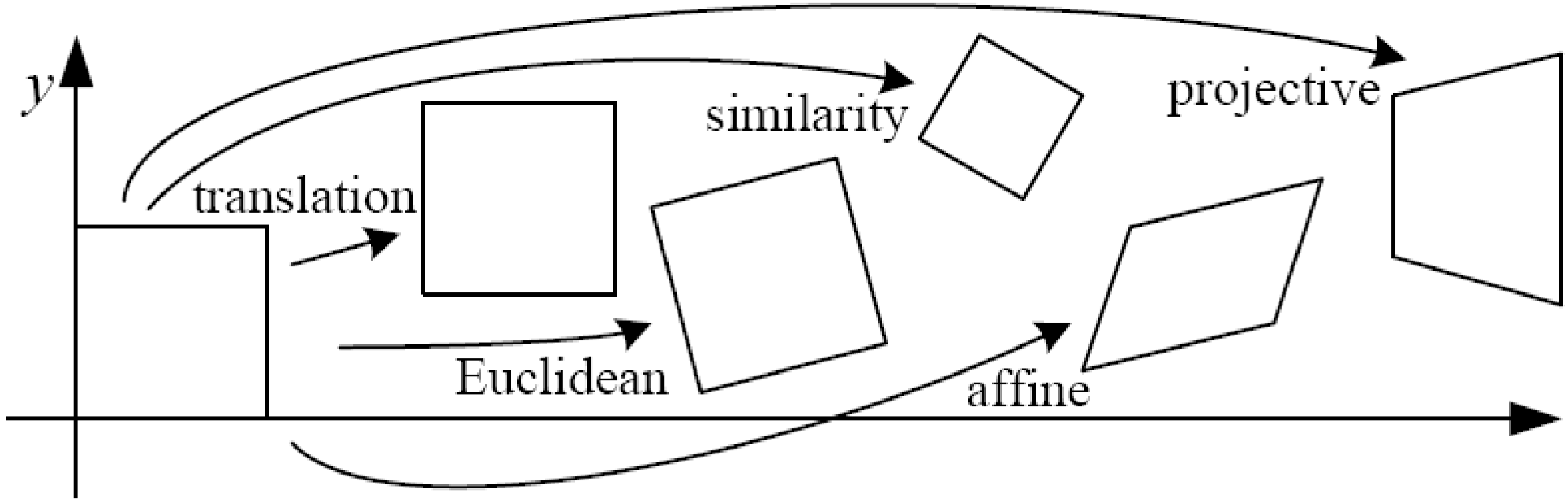
Classification of 2D transformations



Which kind transformation is needed to warp projective plane 1 into projective plane 2?

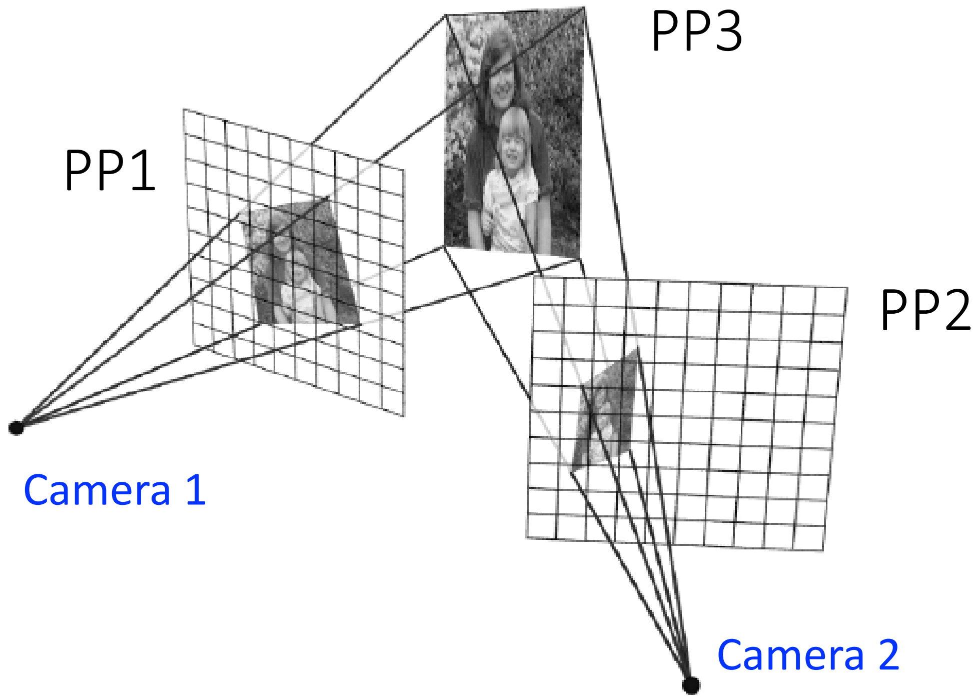


Classification of 2D transformations



Which kind transformation is needed to warp projective plane 1 into projective plane 2?

- A projective transformation (a.k.a. a homography).



Projective transformations

Projective transformations are combinations of

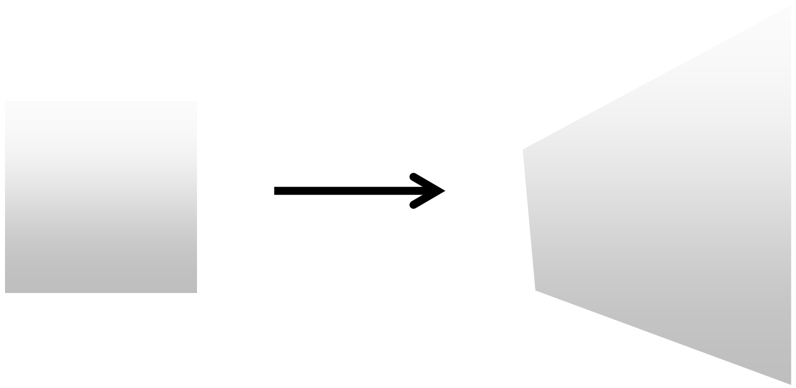
- affine transformations; and
- projective wraps

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

8 DOF: vectors (and therefore matrices) are defined up to scale)

Properties of projective transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines do not necessarily map to parallel lines
- ratios are not necessarily preserved
- compositions of projective transforms are also projective transforms



Computing with homographies

Applying a homography

1. Convert to homogeneous coordinates: $$p = \begin{bmatrix} x \\ y \end{bmatrix} \;\; \Rightarrow \;\; P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

What is the size of the homography matrix?

2. Multiply by the homography matrix: $$P' = H \cdot P$$

3. Convert back to heterogeneous coordinates: $$P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \;\; \Rightarrow \;\; p' = \begin{bmatrix} x'/_{w'} \\ y'/_{w'} \end{bmatrix}$$

Applying a homography

1. Convert to homogeneous coordinates: $p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

What is the size of the homography matrix? Answer: 3 x 3

2. Multiply by the homography matrix: $P' = H \cdot P$

How many degrees of freedom does the homography matrix have?

3. Convert back to heterogeneous coordinates: $P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$

Applying a homography

1. Convert to homogeneous coordinates: $p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

What is the size of the homography matrix? Answer: 3 x 3

2. Multiply by the homography matrix: $P' = H \cdot P$

How many degrees of freedom does the homography matrix have? Answer: 8

3. Convert back to heterogeneous coordinates: $P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/_{w'} \\ y'/_{w'} \end{bmatrix}$

Applying a homography

What is the size of the homography matrix? Answer: 3 x 3

$$P' = H \cdot P$$

How many degrees of freedom does the homography matrix have? Answer: 8

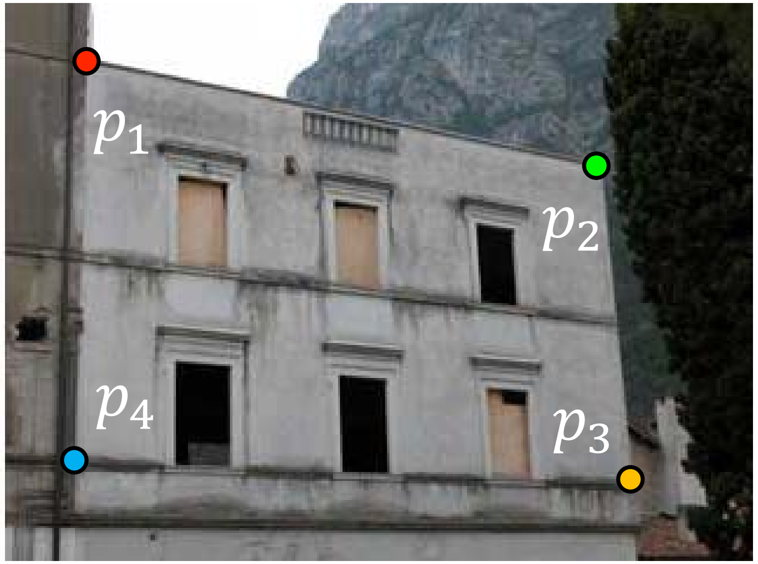
How do we compute the homography matrix?

The direct linear transform (DLT)

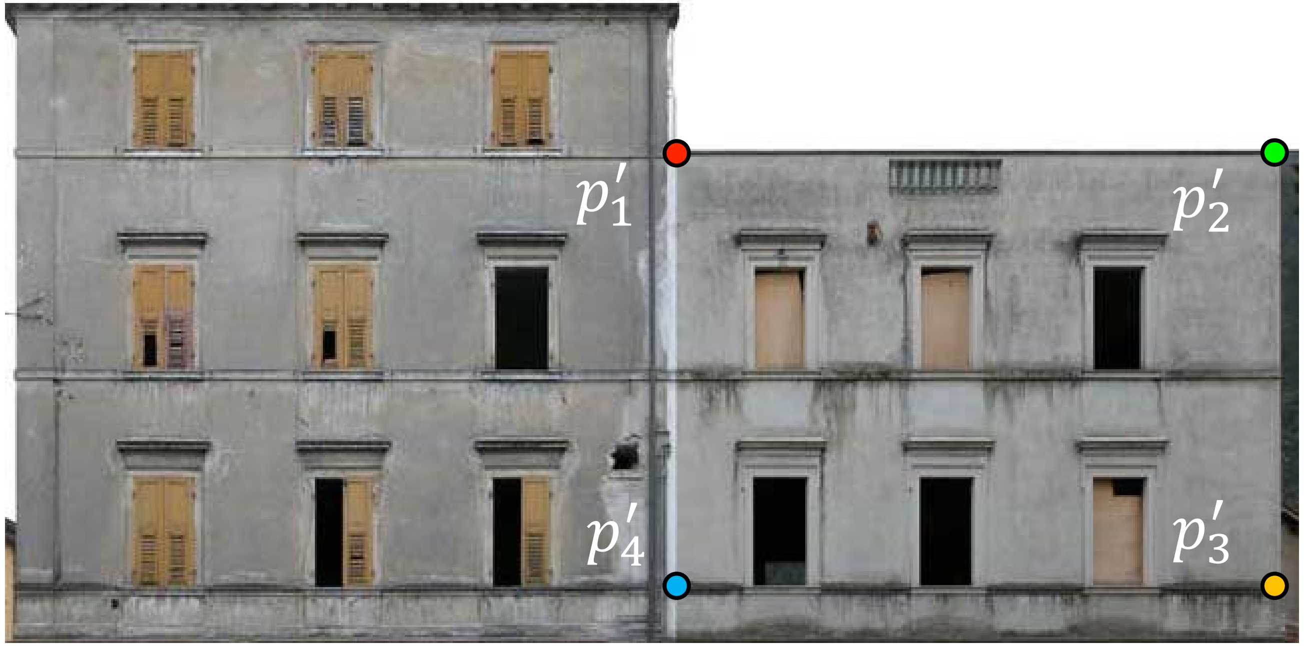
Create point correspondences

Given a set of matched feature points $\{p_i, p_i'\}$ find the best estimate of H such that

$$P' = H \cdot P$$



original image



target image

How many correspondences do we need?

Determining the homography matrix

Write out linear equation for each correspondence:

$$\alpha_j P_j' = H \cdot P_j$$ or $$\alpha_j \begin{bmatrix} x_j' \\ y_j' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix}$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$\alpha_j P_j' = H \cdot P_j \quad \text{or} \qquad \alpha_j \begin{bmatrix} x_j' \\ y_j' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$\alpha_j x_j' = h_1 x_j + h_2 y_j + h_3$$
$$\alpha_j y_j' = h_4 x_j + h_5 y_j + h_6$$
$$\alpha_j = h_7 x_j + h_8 y_j + h_9$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$\alpha_j P_j' = H \cdot P_j \quad \text{or} \qquad \alpha_j \begin{bmatrix} x_j' \\ y_j' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$\alpha_j x_j' = h_1 x_j + h_2 y_j + h_3$$
$$\alpha_j y_j' = h_4 x_j + h_5 y_j + h_6$$
$$\alpha_j = h_7 x_j + h_8 y_j + h_9$$

Divide out unknown scale factor:

$$x_j'(h_7 x_j + h_8 y_j + h_9) = h_1 x_j + h_2 y_j + h_3$$
$$y_j'(h_7 x_j + h_8 y_j + h_9) = h_4 x_j + h_5 y_j + h_6$$

How do you rearrange terms to make it a linear system?

$$x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3)$$
$$y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6)$$

Just rearrange the terms

$$\boxed{\begin{aligned} h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0 \\ h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0 \end{aligned}}$$

Determining the homography matrix

Re-arrange terms:

$$h_7 x x' + h_8 y x' + h_9 x' - h_1 x - h_2 y - h_3 = 0$$
$$h_7 x y' + h_8 y y' + h_9 y' - h_4 x - h_5 y - h_6 = 0$$

Re-write in matrix form:

How many equations from one point correspondence?

$$\mathbf{A}_i \boldsymbol{h} = \mathbf{0}$$

$$\mathbf{A}_i = \left[\begin{array}{ccccccccc} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{array} \right]$$

$$\boldsymbol{h} = \left[\begin{array}{ccccccccc} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{array} \right]^\top$$

Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{array}{l} \text{Corresponding point pair 1} \\ \\ \text{Corresponding point pair 2} \\ \\ \\ \text{Corresponding point pair j} \end{array}
\begin{array}{c}
\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\
\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\
\vdots \\
\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}
\end{array}
\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

Reminder: Determining unknown transformations

Affine transformation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Why can we drop the last line?

Vectorize transformation parameters:

Stack equations from point correspondences:

$$\underbrace{\begin{bmatrix} x' \\ y' \\ x' \\ y' \\ \vdots \\ x' \\ y' \end{bmatrix}}_{\boldsymbol{b}} = \underbrace{\begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ & & & \vdots & & \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}}_{\boldsymbol{x}}$$

Notation in system form:

$$\boxed{Ax = b}$$

Reminder: Determining unknown transformations

Convert the system to a linear least-squares problem:

$$E_{\mathrm{LLS}} = \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2$$

Expand the error:

$$E_{\mathrm{LLS}} = \boldsymbol{x}^\top(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} - 2\boldsymbol{x}^\top(\mathbf{A}^\top\boldsymbol{b}) + \|\boldsymbol{b}\|^2$$

Minimize the error:

Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b}$

Solve for x $\boldsymbol{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$ ⟵

In Matlab:

```
x = A \ b
```

Note: You almost <u>never</u> want to compute the inverse of a matrix.

Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \vdots \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

- How do we solve this?

Determining the homography matrix

Stack together constraints from multiple point correspondences:

$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \vdots \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

- Solve with SVD

General form of total least squares

(**Warning:** change of notation. x is a vector of parameters!)

$$E_{\mathrm{TLS}} = \sum_i (\boldsymbol{a}_i \boldsymbol{x})^2$$

$$= \|\mathbf{A}\boldsymbol{x}\|^2$$ (matrix form)

$$\|\boldsymbol{x}\|^2 = 1$$ constraint

minimize $\|\mathbf{A}\boldsymbol{x}\|^2$

subject to $\|\boldsymbol{x}\|^2 = 1$

$\Rightarrow$

minimize $\dfrac{\|\mathbf{A}\boldsymbol{x}\|^2}{\|\boldsymbol{x}\|^2}$ (Rayleigh quotient)

Solution is the eigenvector corresponding to smallest eigenvalue of

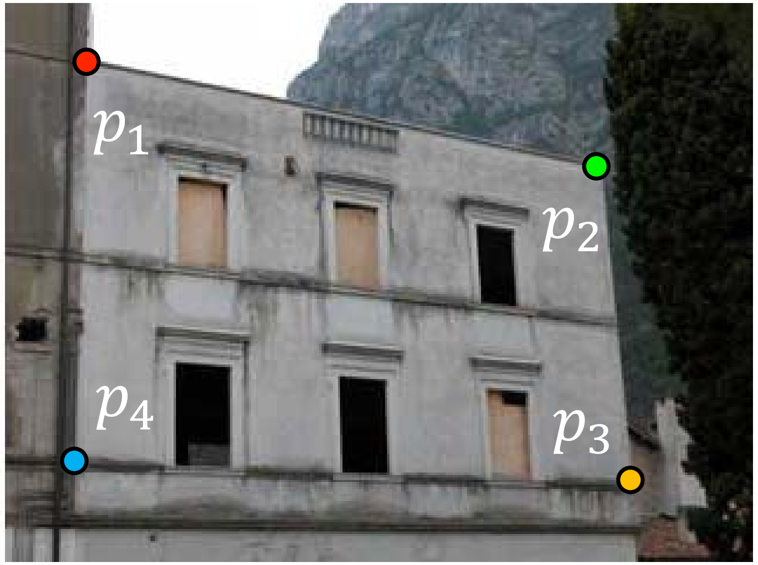
$$\mathbf{A}^\top \mathbf{A}$$

(equivalent)

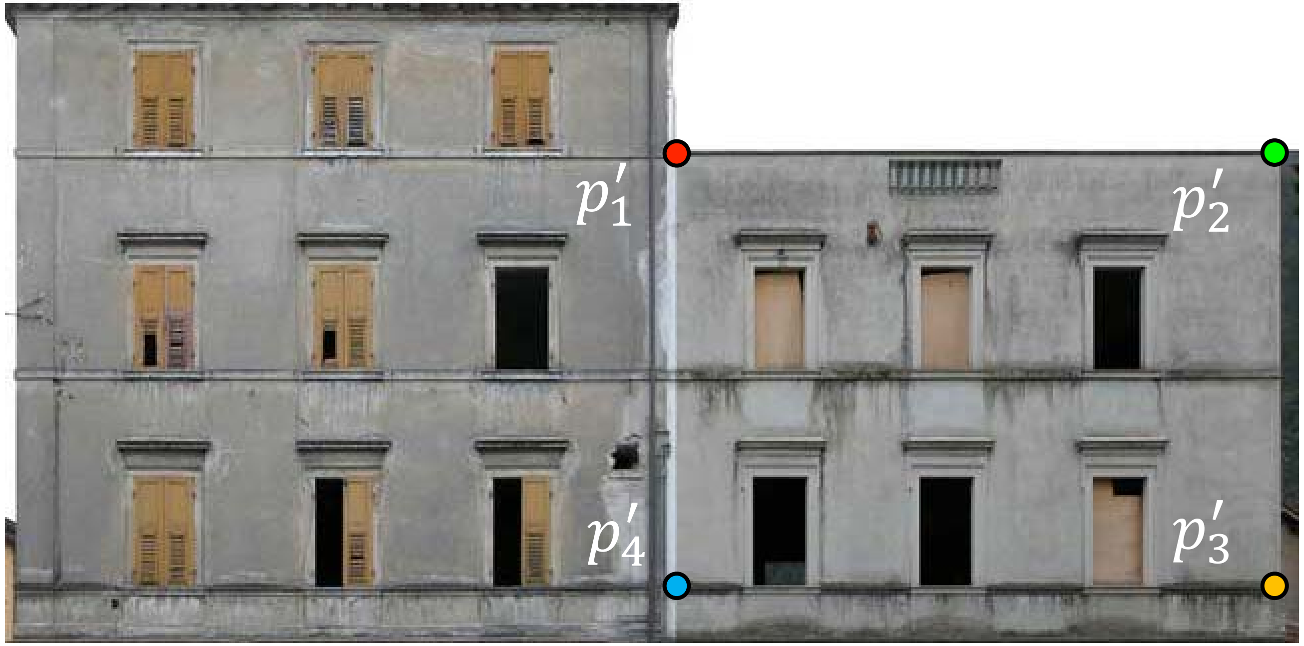
Solution is the column of **V** corresponding to smallest singular value

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

Create point correspondences



original image



target image

How many corresponding points you need?

Constraining homography estimation

2 equations from each corresponding pairs

$$h_7 x x' + h_8 y x' + h_9 x' - h_1 x - h_2 y - h_3 = 0$$
$$h_7 x y' + h_8 y y' + h_9 y' - h_4 x - h_5 y - h_6 = 0$$

Homography has 8 degrees of freedom (9 elements up to scale)

Need $\geq$ 4 corresponding points

If you use $\geq$ 4 noise free corresponding points (which arise from the same planes) you still have a rank 8 A matrix

Linear least squares estimation only works when the transform function is ?

Linear least squares estimation only works when the transform function is **linear! (duh)**

Also doesn’t deal well with outliers

Problems with linearization

Started with: find H such that for all points P_j

$$P_j' \approx \alpha H \cdot P_j$$

$$P_j = \begin{pmatrix} x_j \\ y_j \\ 1 \end{pmatrix} \quad P_j' = \begin{pmatrix} x_j' \\ y_j' \\ 1 \end{pmatrix}$$

$$H = \begin{bmatrix} -H_1- \\ -H_2- \\ -H_3- \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

What we are minimizing

$$\sum_j \quad (H_1 P_j - H_3 P_j')^2 + (H_2 P_j - H_3 P_j')^2$$

$$(h_7 x_j x_j' + h_8 y_j x_j' + h_9 x_j' - h_1 x_j - h_2 y_j - h_3)^2 +$$
$$(h_7 x_j y_j' + h_8 y_j y_j' + h_9 y_j' - h_4 x_j - h_5 y_j - h_6)^2$$

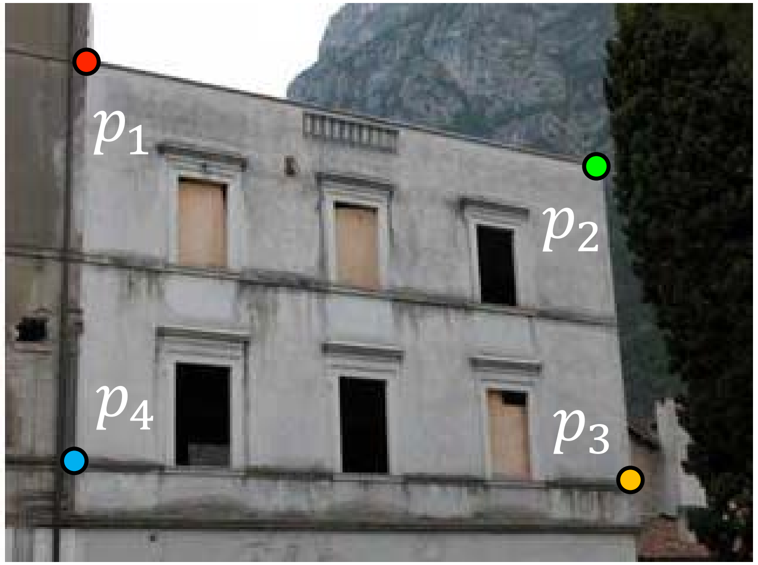
$$\neq$$

Given localization errors this is NOT the same

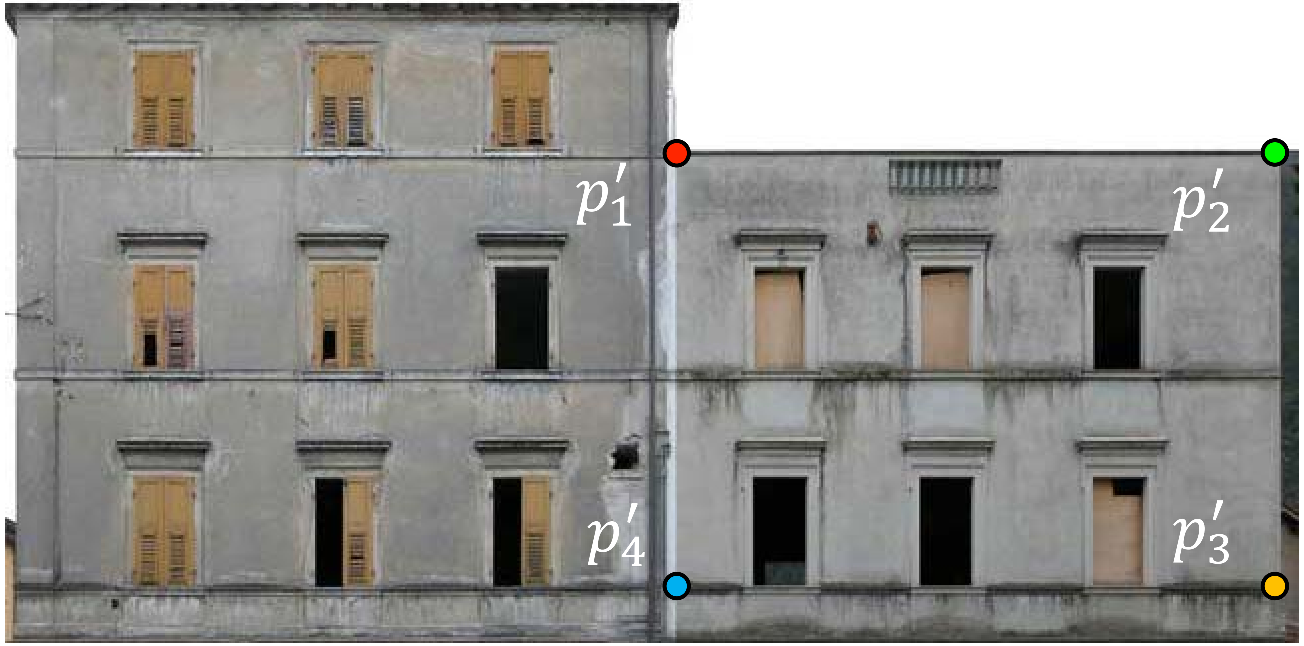
What do we want to minimize

$$\sum_j \left(\frac{H_1 P_j}{H_3 P_j} - P_{j,1}' \right)^2 + \left(\frac{H_2 P_j}{H_3 P_j'} - P_{j,2}' \right)^2$$

Create point correspondences



original image



target image

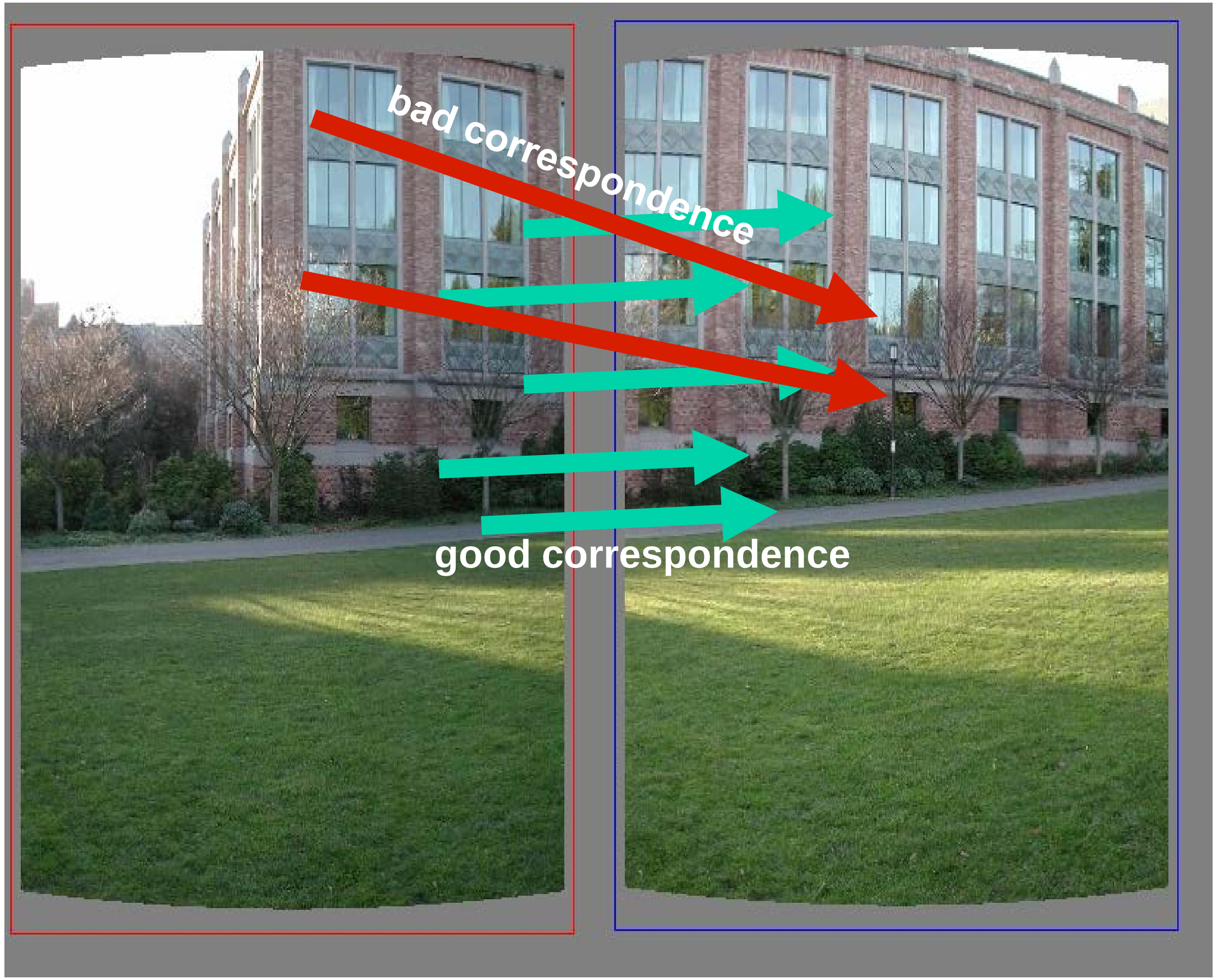
How do we automate this step?

The image correspondence pipeline

1. Feature point detection
   - Detect corners using the Harris corner detector.

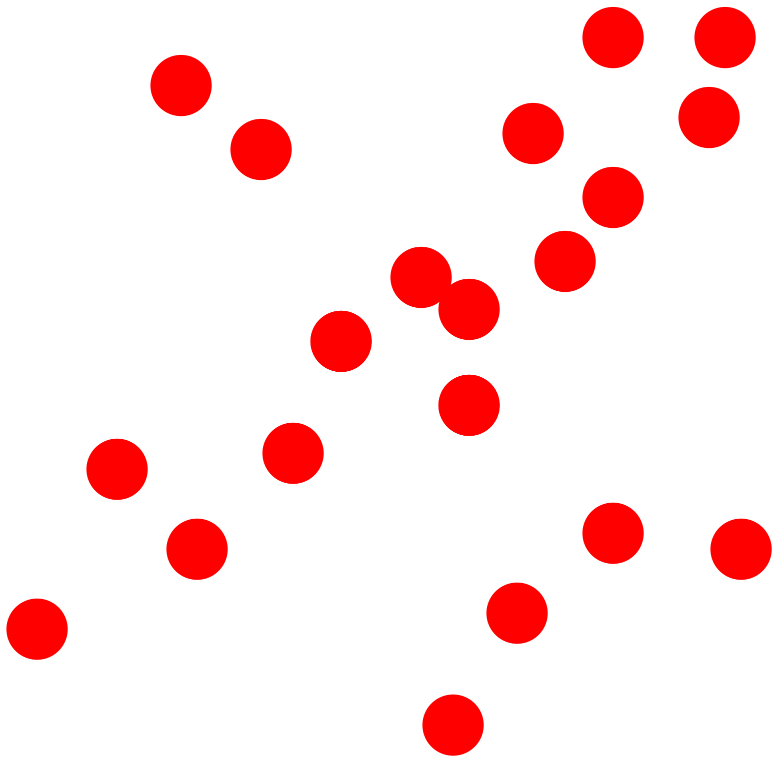
2. Feature point description
   - Describe features using the Multi-scale oriented patch descriptor. (e.g. SIFT)

3. Feature matching


bad correspondence
good correspondence

Random Sample Consensus (RANSAC)

Fitting lines
(with outliers)

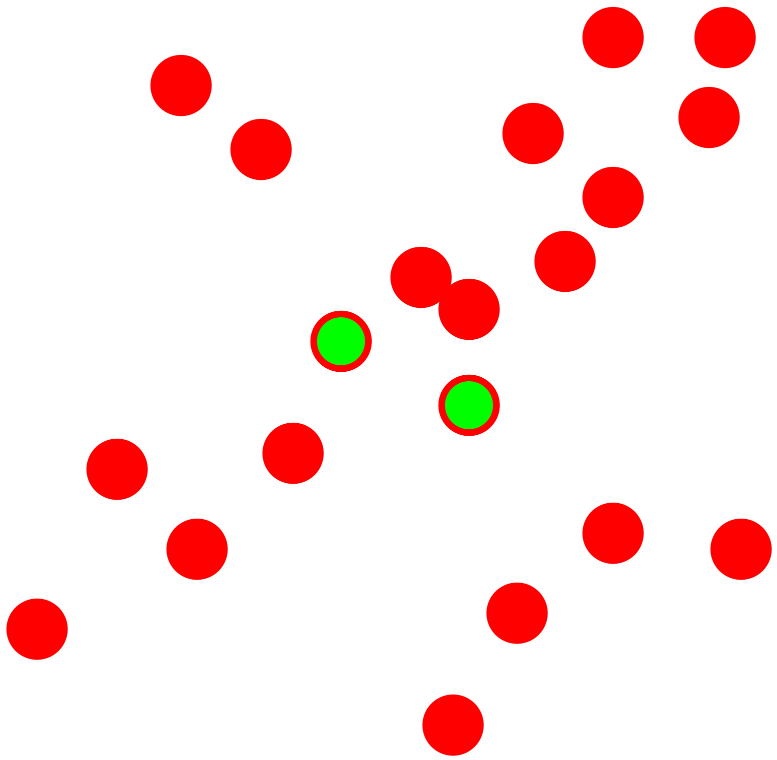


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

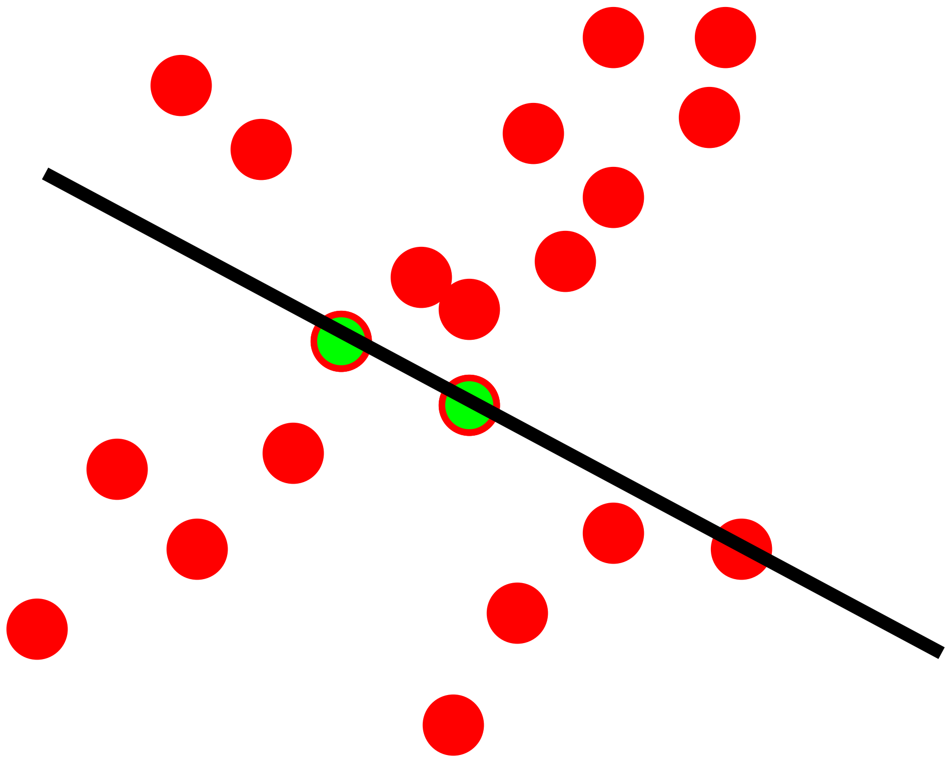


Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

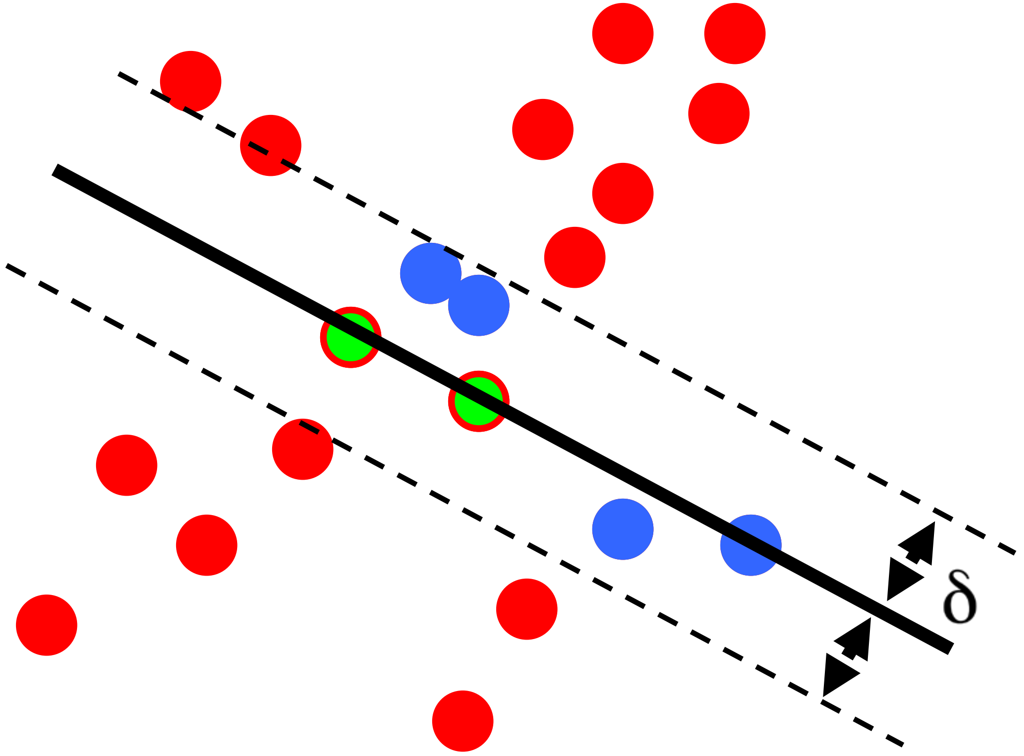


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



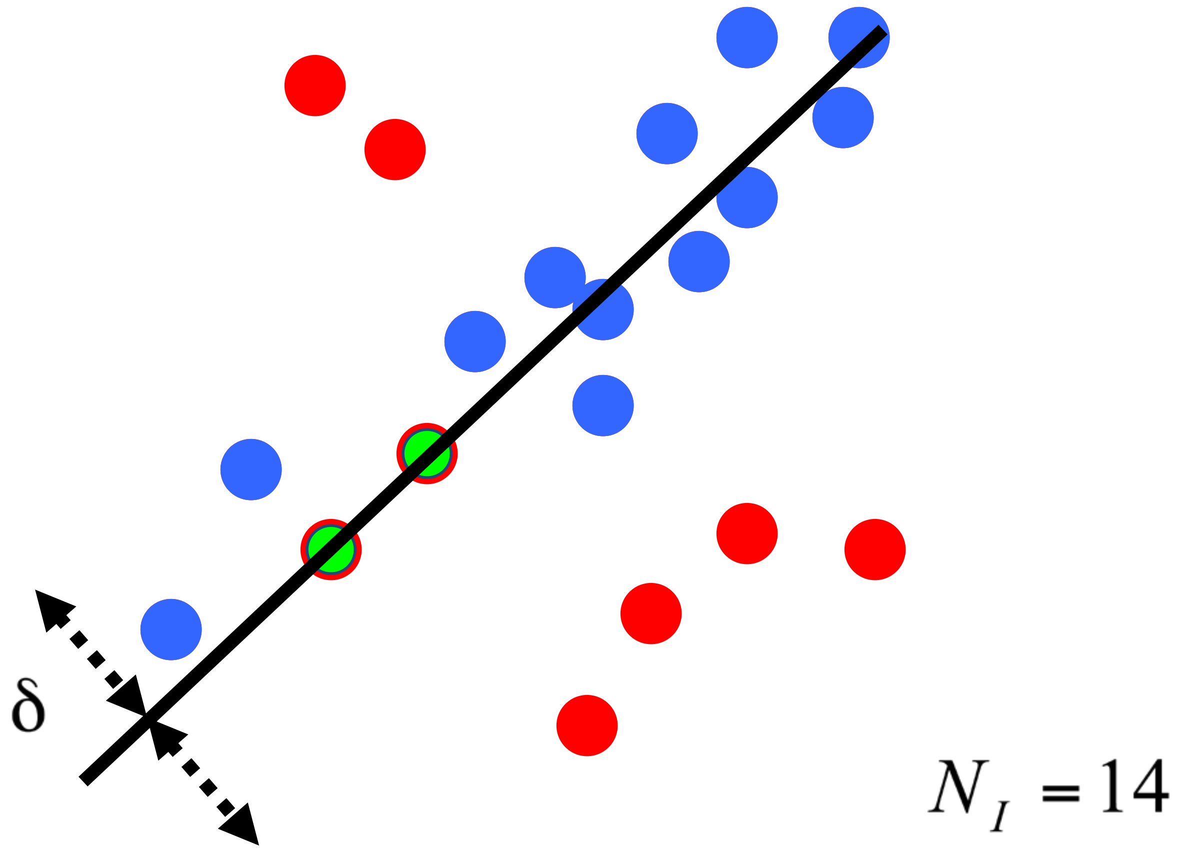
$N_I = 6$

Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e)
- Number of sampled points s
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

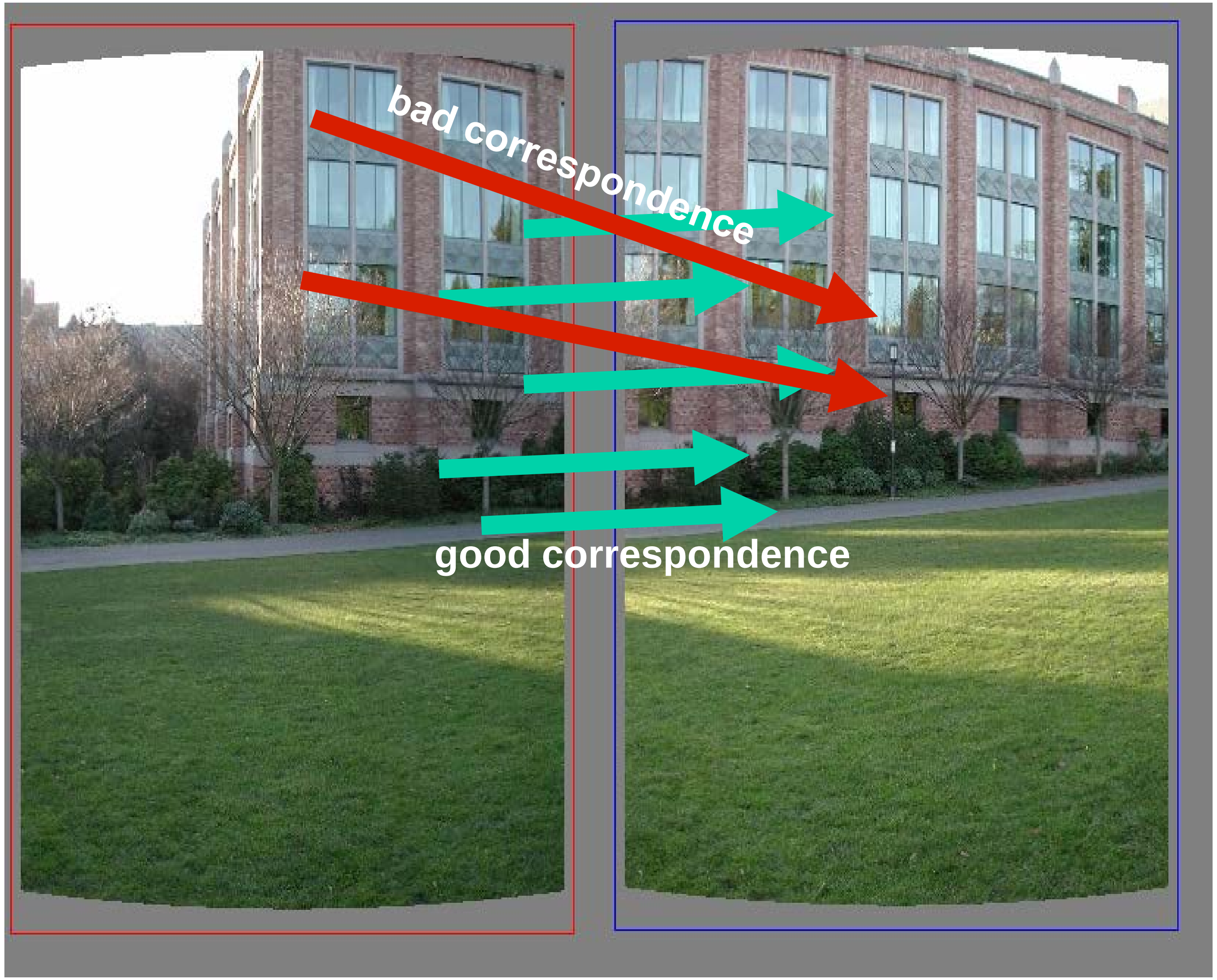
$$N = \frac{\log(1-p)}{\log\left(1-(1-e)^s\right)}$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

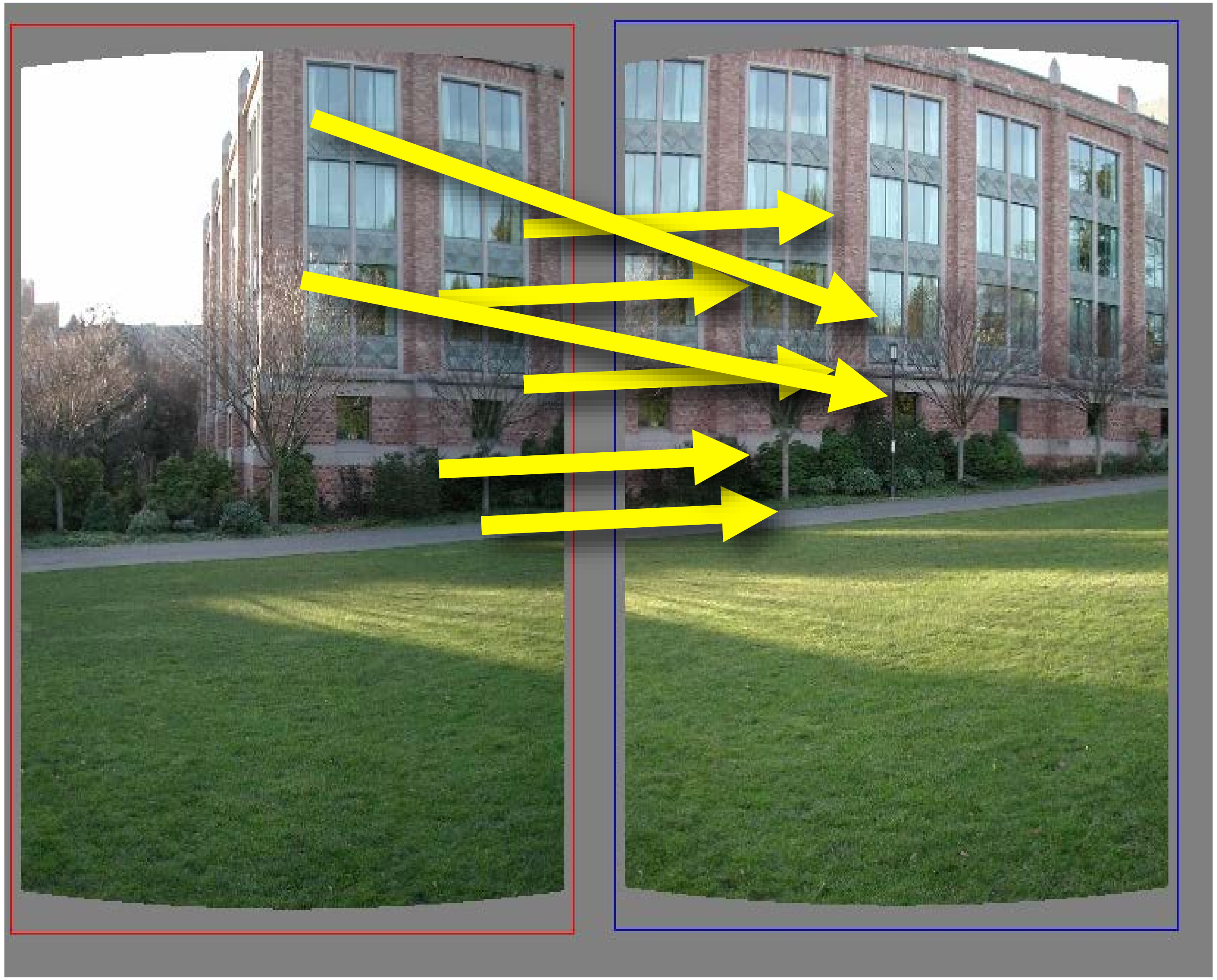
Given two images…

find matching features (e.g., SIFT) and a translation transform

Matched points will usually contain bad correspondences

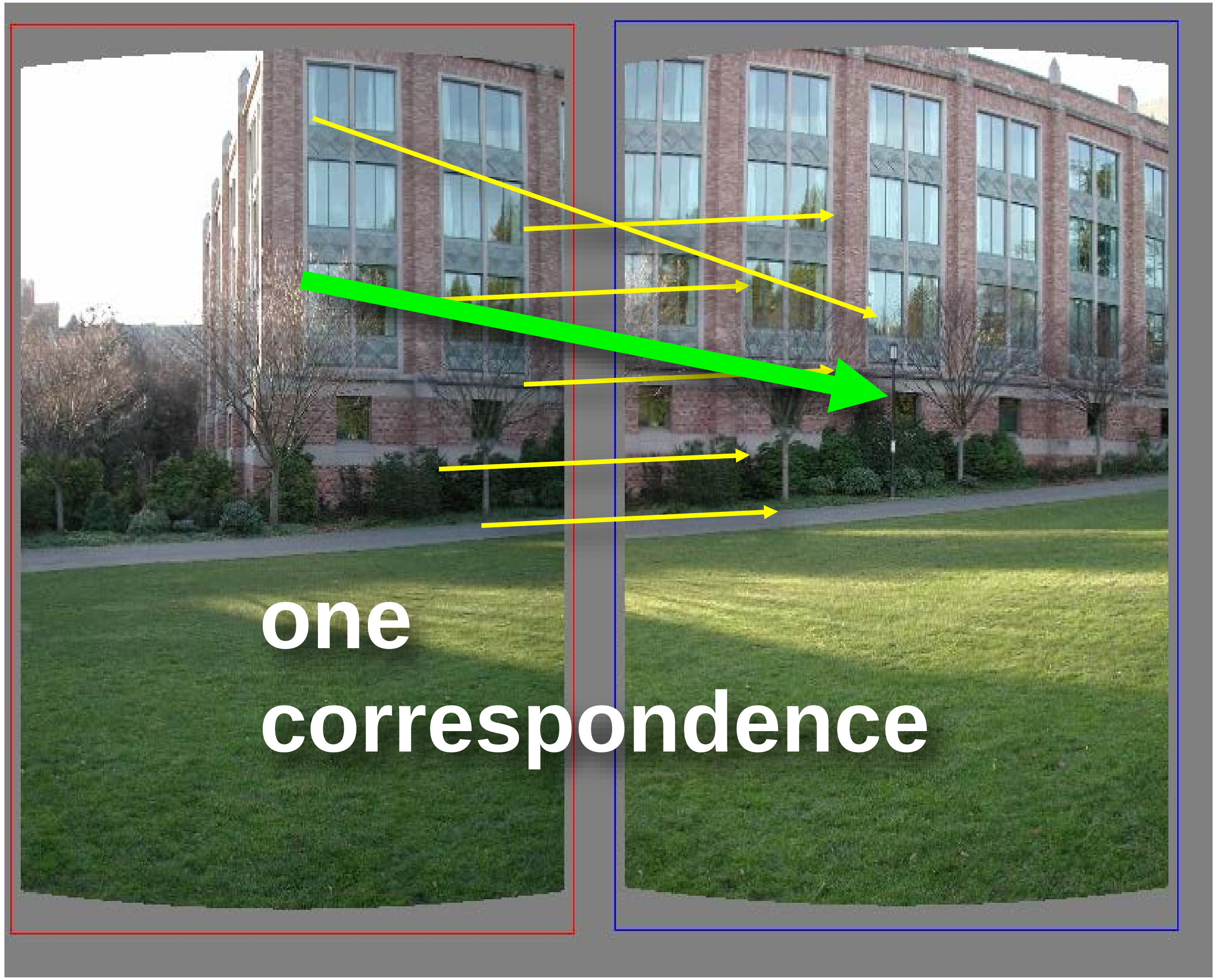


how should we estimate the transform?

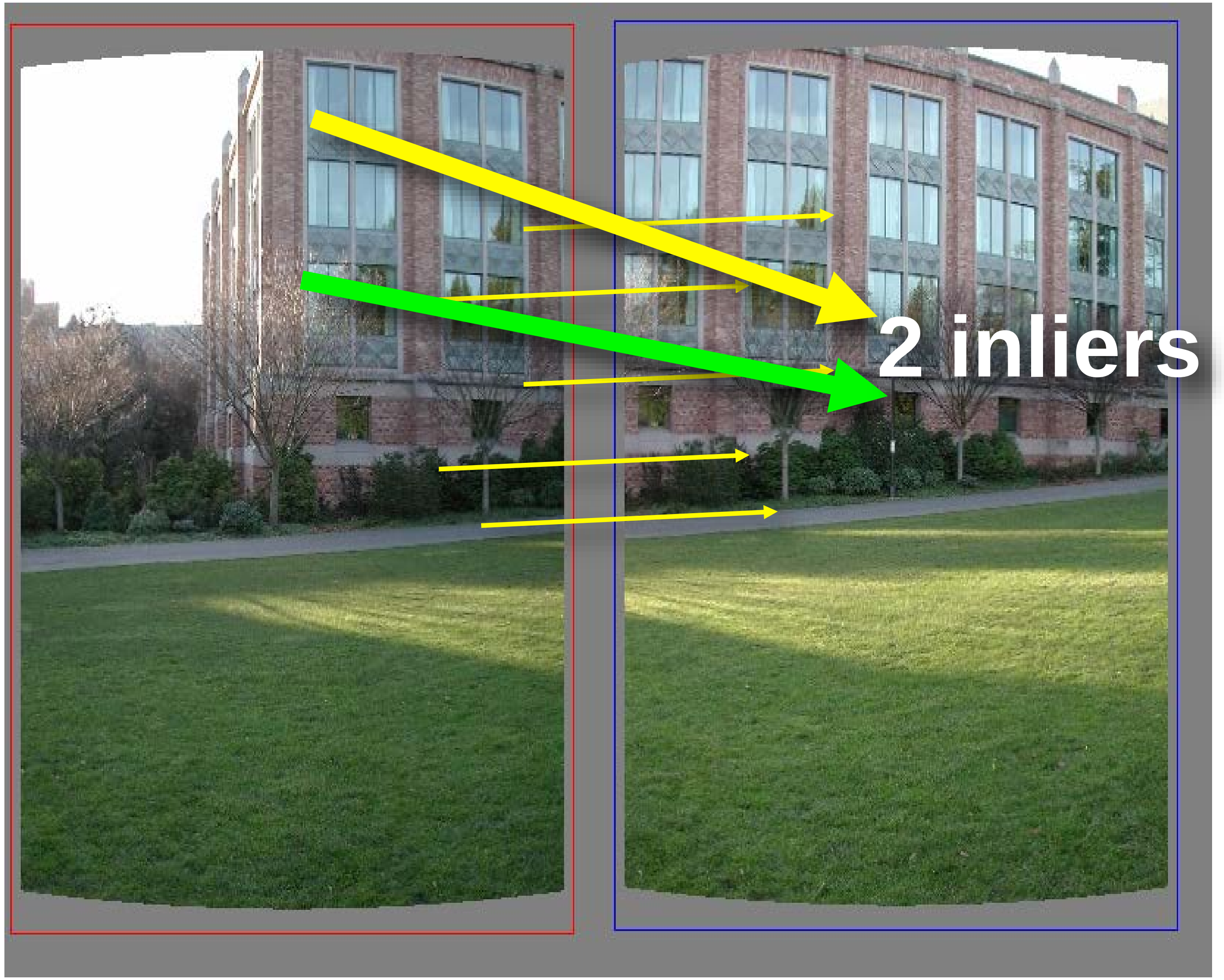


Need only **one correspondence**, to find translation model

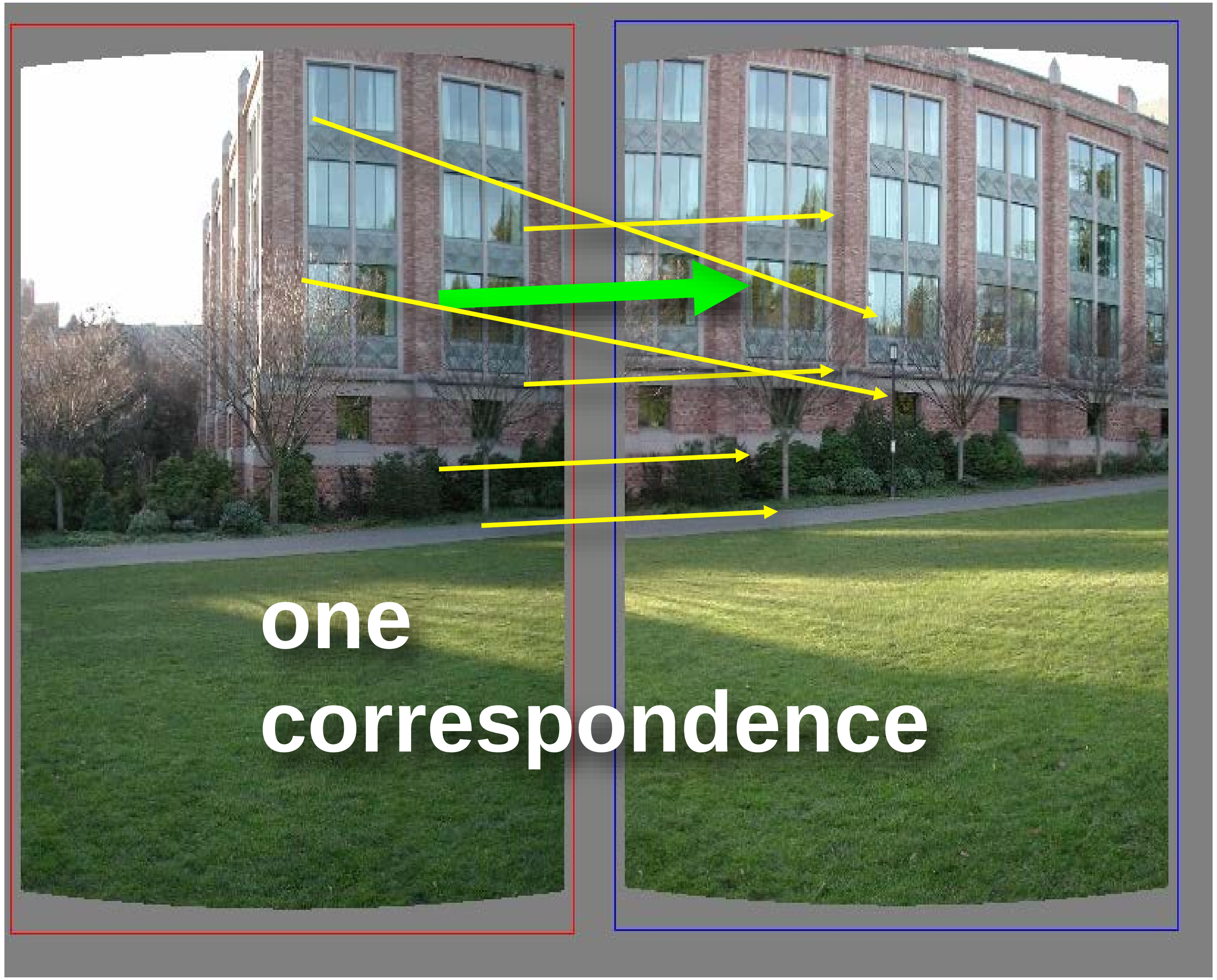
Pick one correspondence, count inliers



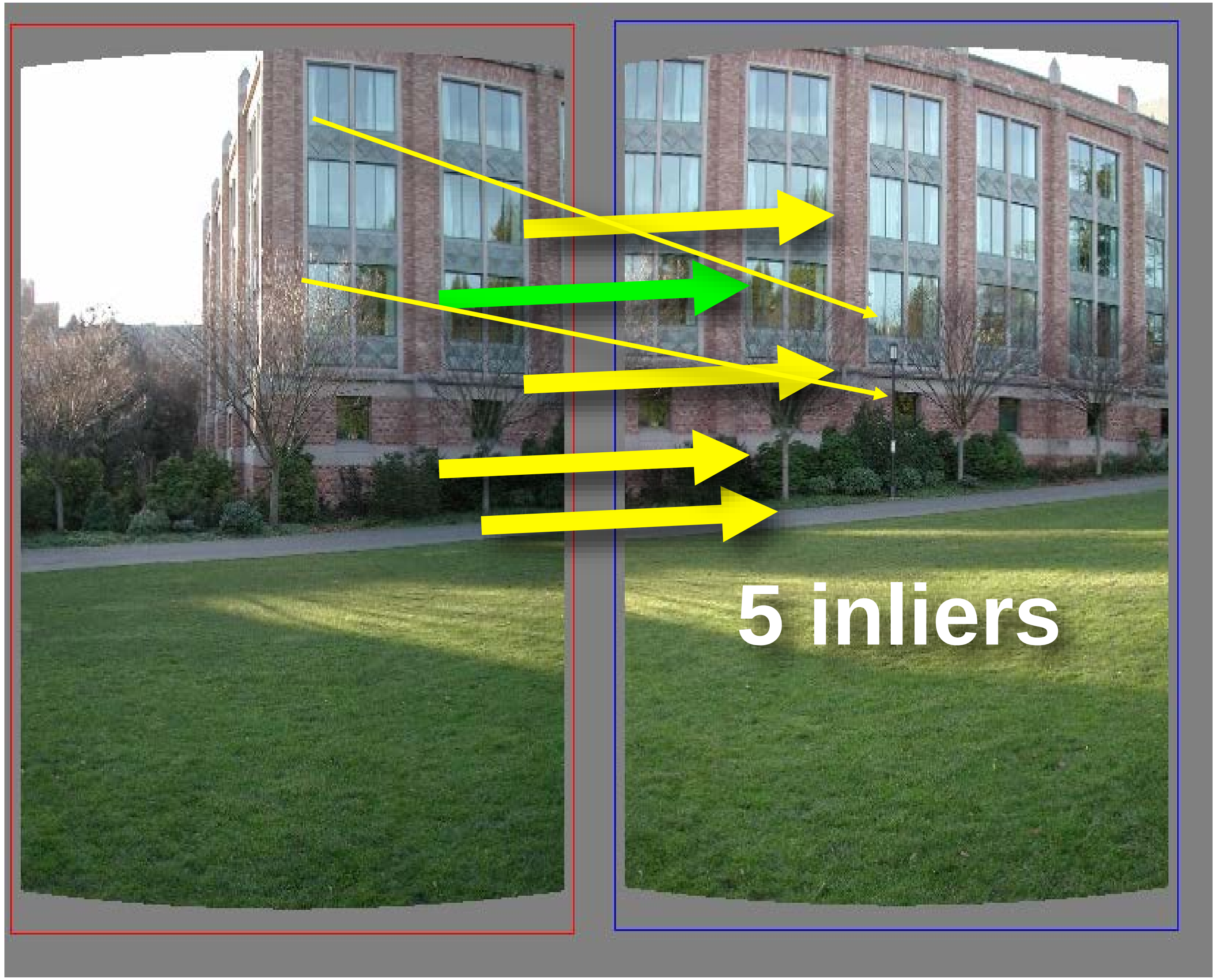
Pick one correspondence, count inliers



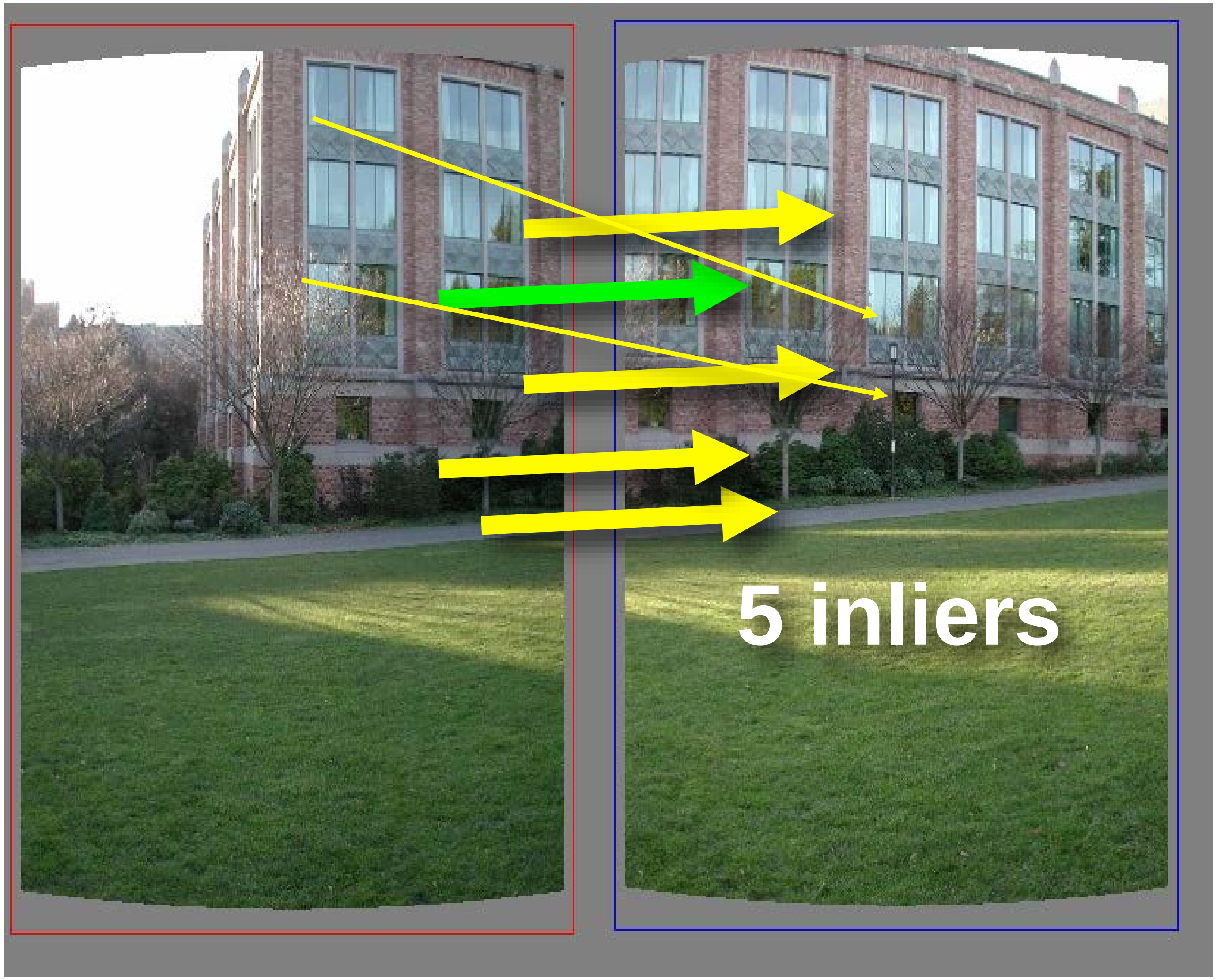
Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick the model with the highest number of inliers!

Estimating homography using RANSAC

- RANSAC loop

  1. Get point correspondences (randomly)

Estimating homography using RANSAC

- RANSAC loop

  1. Get four point correspondences (randomly)
  2. Compute H using

Estimating homography using RANSAC

- RANSAC loop

1. Get four point correspondences (randomly)
2. Compute H using DLT
3. Count

Estimating homography using RANSAC

- RANSAC loop

  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers
  4. Keep H if

Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers
  4. Keep H if largest number of inliers
- Recompute H using all inliers

The image correspondence pipeline

1. Feature point detection
   - Detect corners using the Harris corner detector.

2. Feature point description
   - Describe features using the Multi-scale oriented patch descriptor.

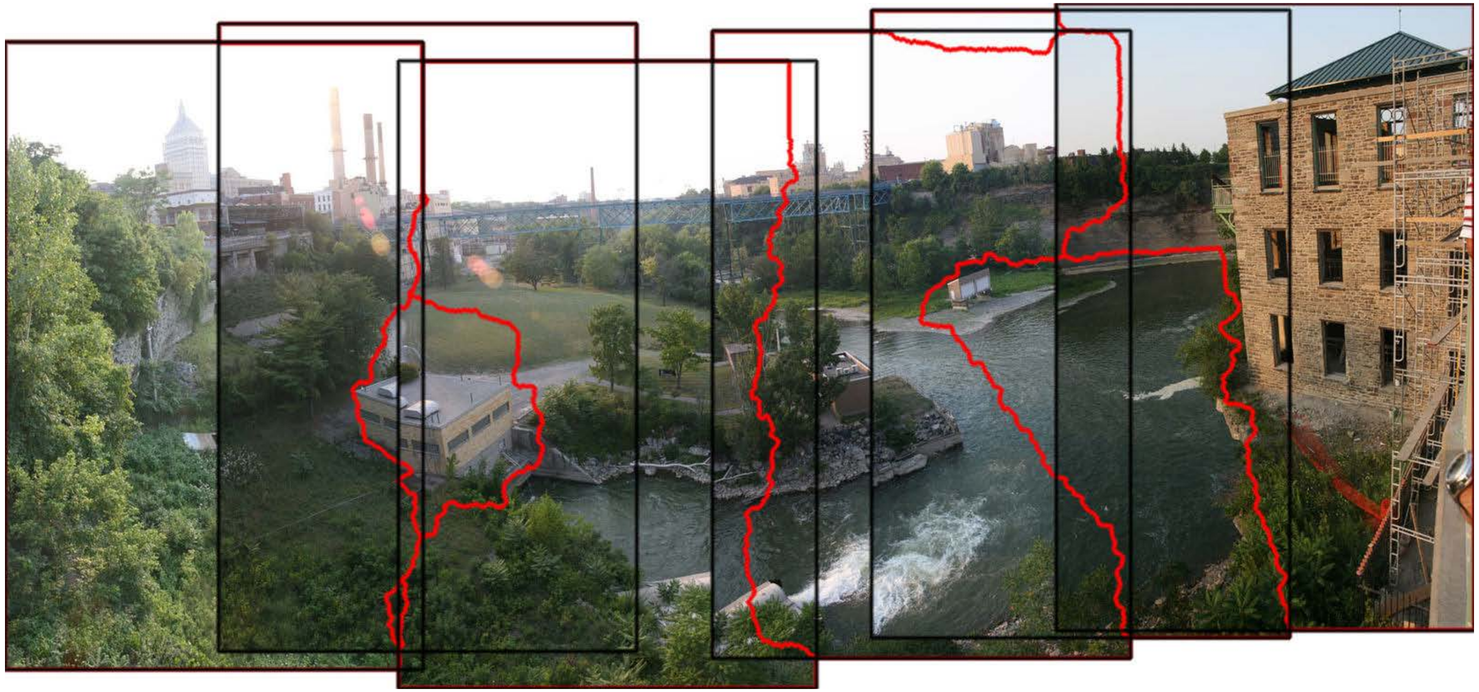
3. Feature matching *and* homography estimation
   - Do both simultaneously using RANSAC.

Panoramas from image stitching

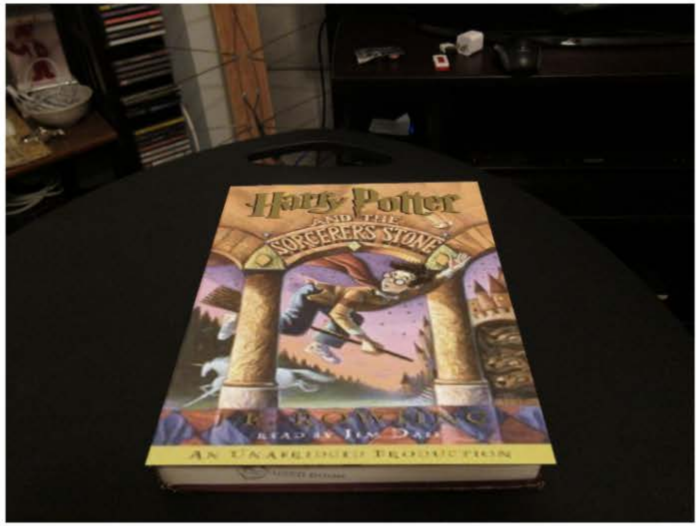
1. Capture multiple images from different viewpoints.

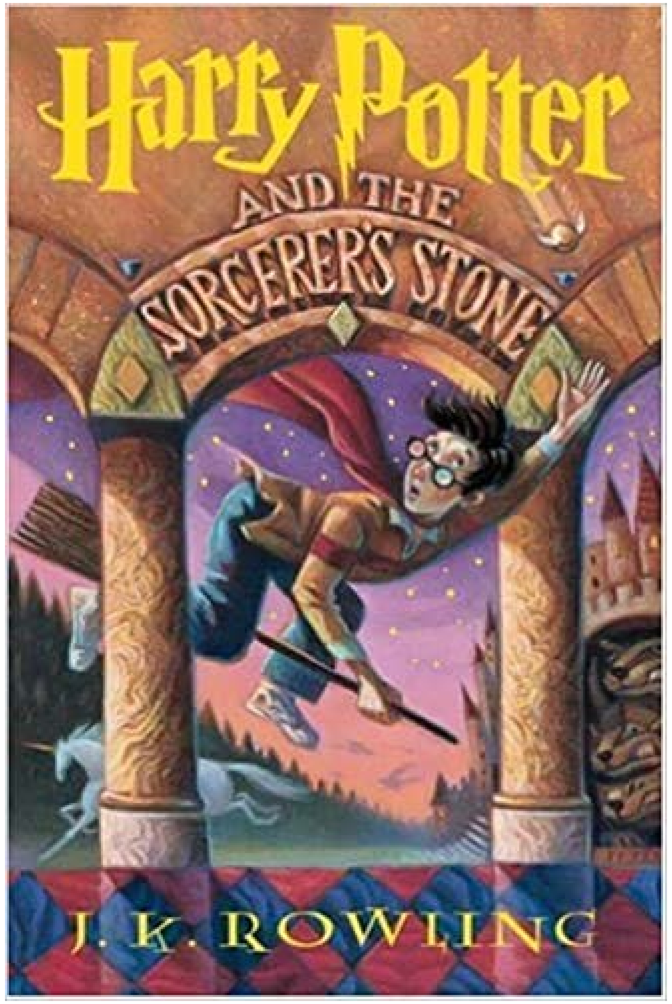
2. Stitch them together into a virtual wide-angle image.

Optional: find a good seam



When can we use homographies?





Theorem: Every image of a ***plane*** in camera 1 can be mapped into an image of a plane in camera 2 via a 3x3 projective homograpy

We can use homographies when…

1. … the scene is planar; or

2. … the scene is very far or has small (relative) depth variation → scene is approximately planar

We can use homographies when…

3. … the scene is captured under camera rotation only (no translation or pose change)

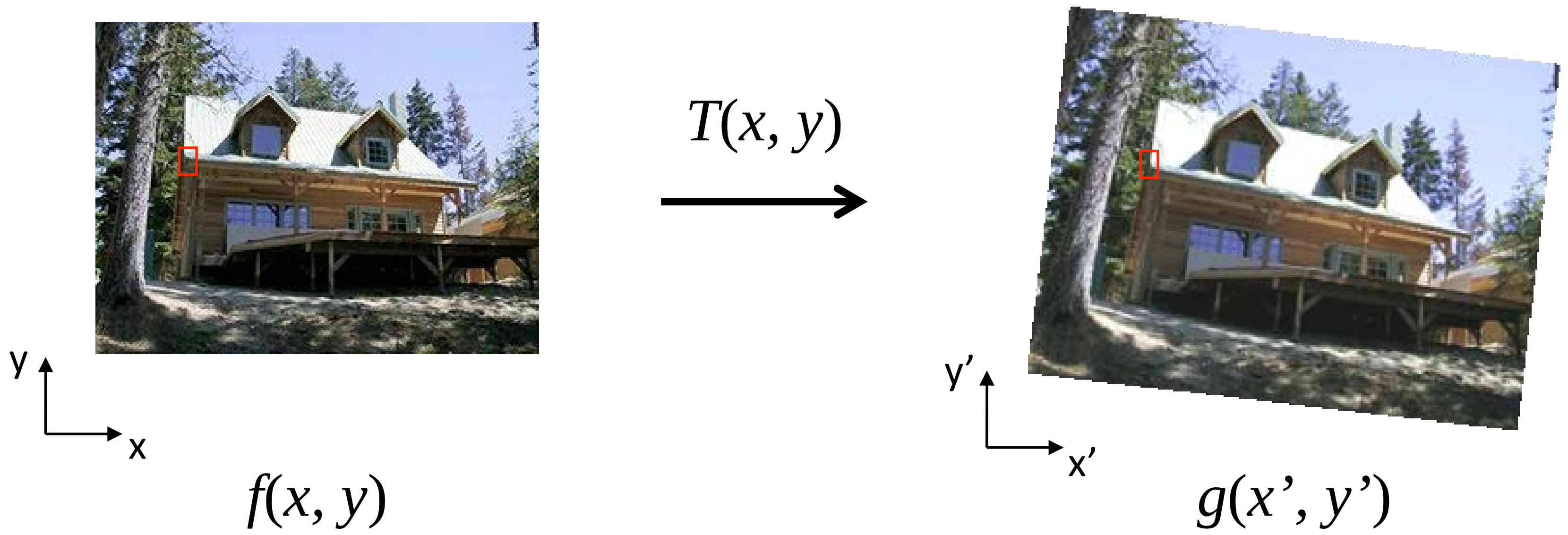
More on why this is the case in a later lecture.

Determining unknown image warps

Determining unknown image warps

Suppose we have two images.

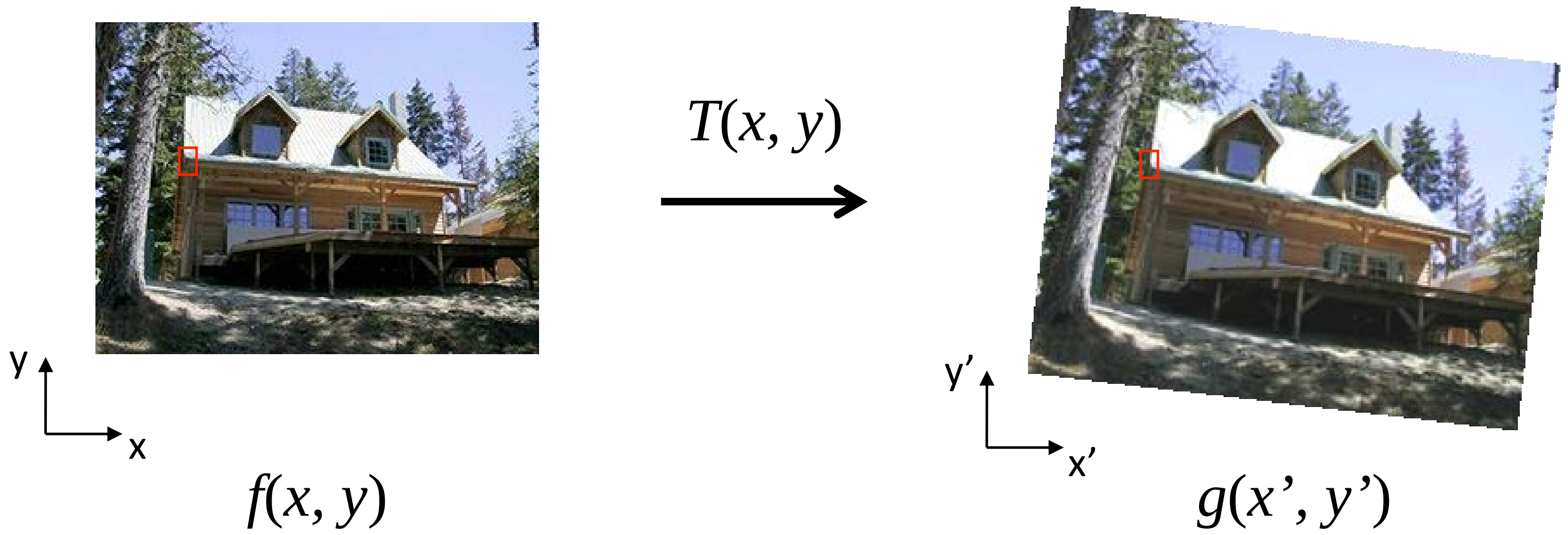
- How do we compute the transform that takes one to the other?



Forward warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?



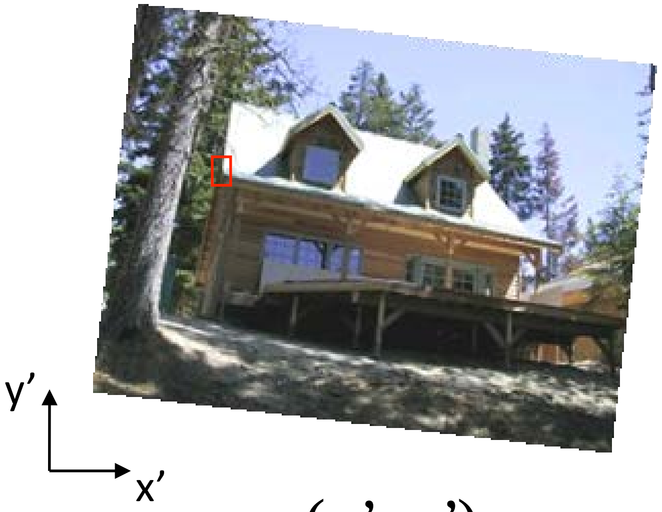
1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before
3. Send intensities $f(x,y)$ in first image to their corresponding location in the second image

Forward warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?

$T(x, y)$



$f(x, y)$

$g(x', y')$

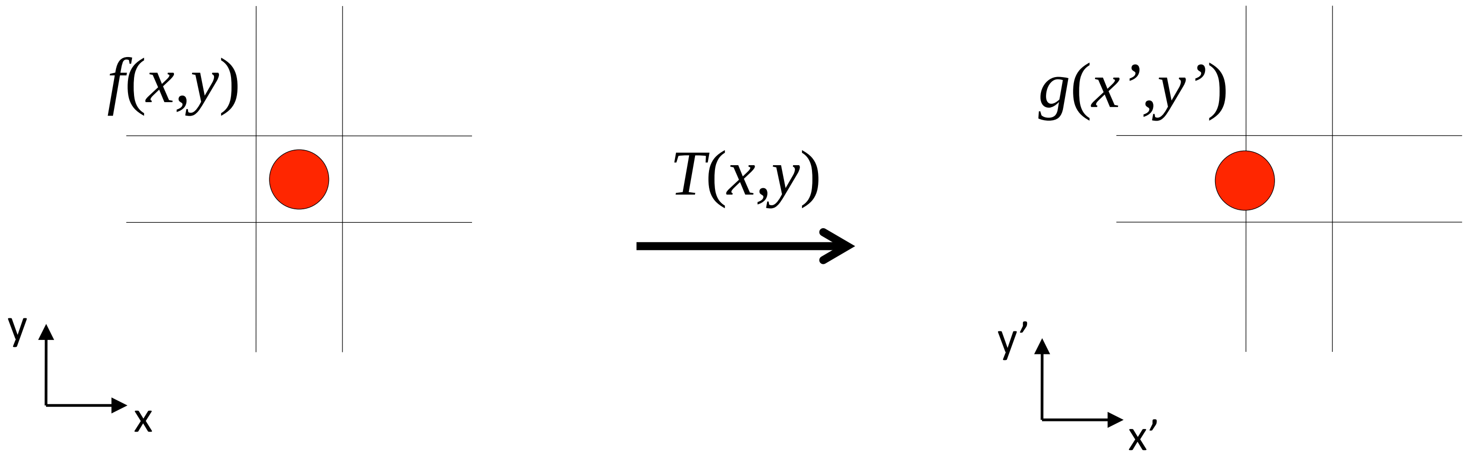
what is the problem with this?

1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before
3. Send intensities $f(x,y)$ in first image to their corresponding location in the second image

Forward warping

Pixels may end up between two points

- How do we determine the intensity of each point?

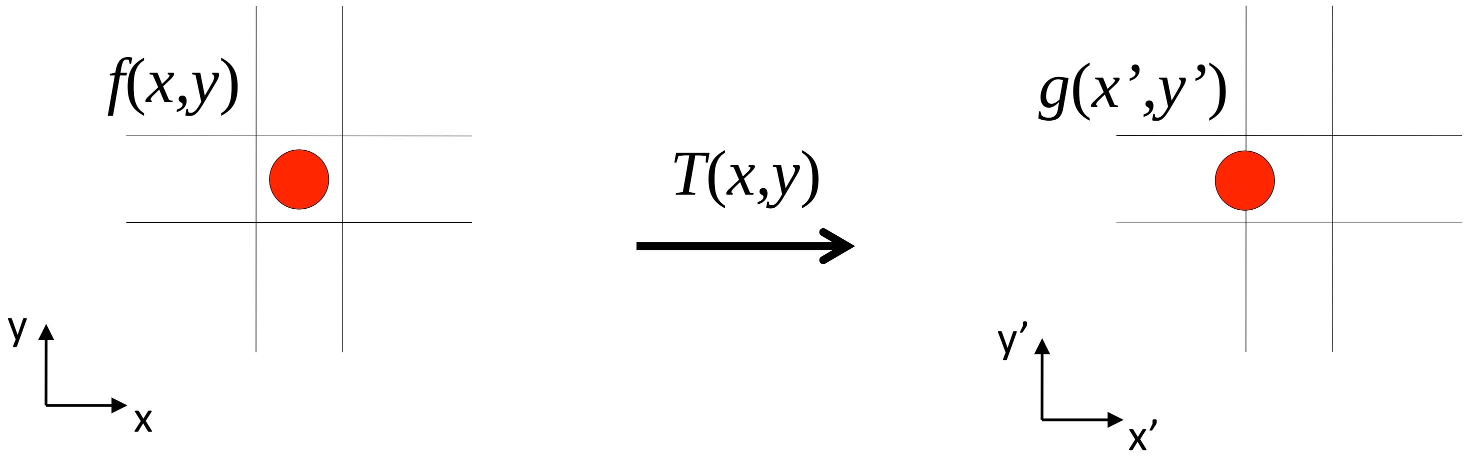


Forward warping

Pixels may end up between two points

- How do we determine the intensity of each point?

✓ We distribute color among neighboring pixels (x’,y’) (“splatting”)



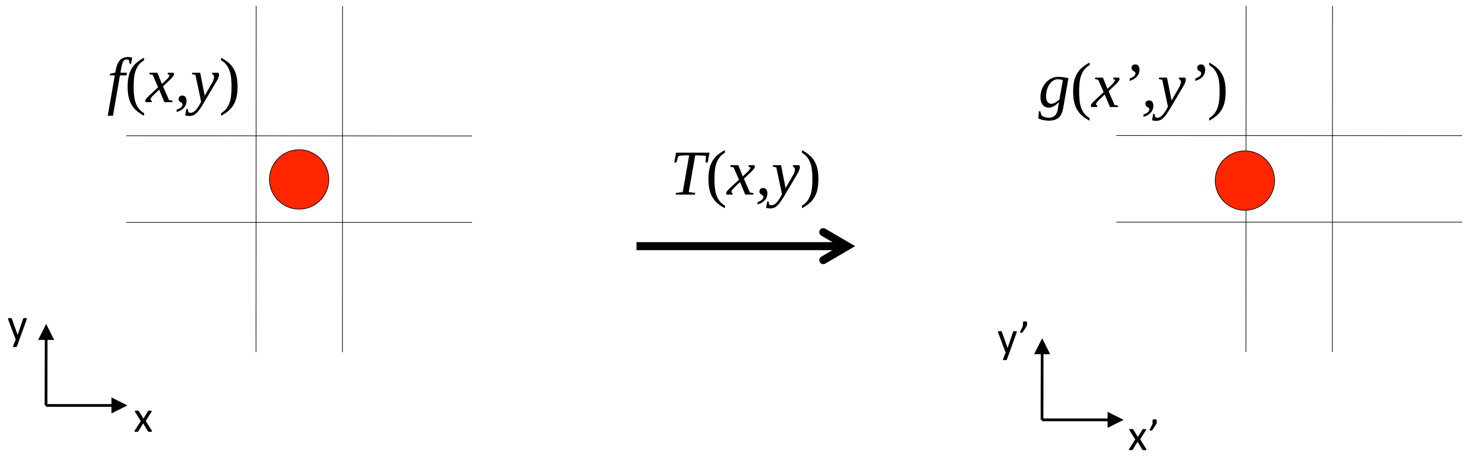
- What if a pixel (x’,y’) receives intensity from more than one pixels (x,y)?

Forward warping

Pixels may end up between two points

- How do we determine the intensity of each point?

✓ We distribute color among neighboring pixels (x',y') ("splatting")



- What if a pixel (x',y') receives intensity from more than one pixels (x,y)?

✓ We average their intensity contributions.

Inverse warping

Suppose we have two images.
- How do we compute the transform that takes one to the other?

$T^{-1}(x, y)$

$\longleftarrow$

$f(x, y)$ $\qquad$ $g(x', y')$

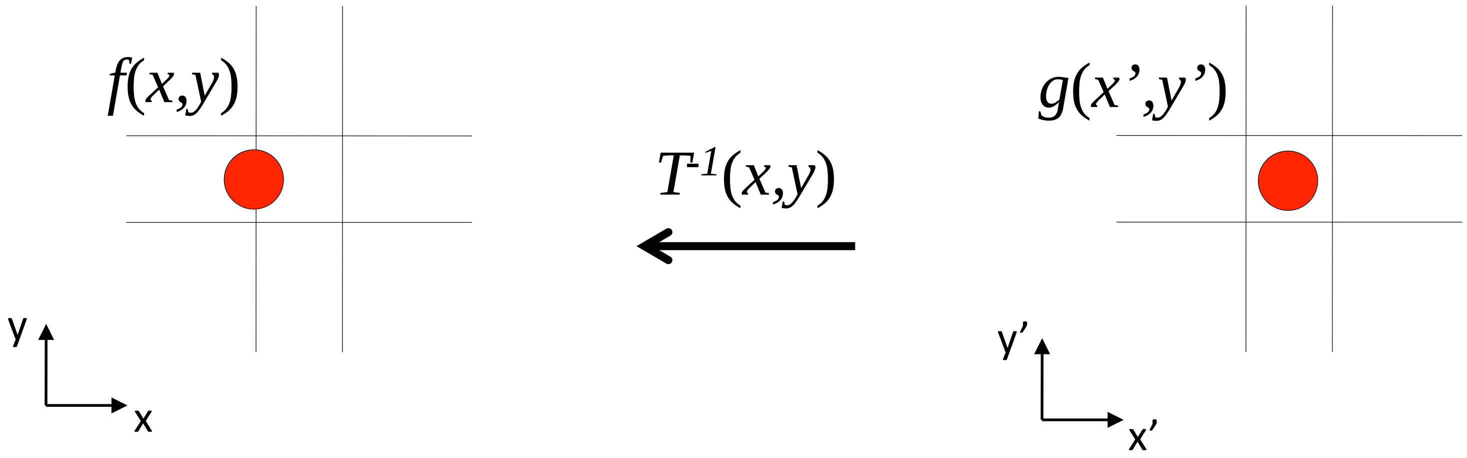
what is the problem with this?

1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before, then compute its inverse
3. Get intensities $g(x',y')$ in in the second image from point $(x,y) = T^{-1}(x',y')$ in first image

Inverse warping

Pixel may come from between two points

- How do we determine its intensity?

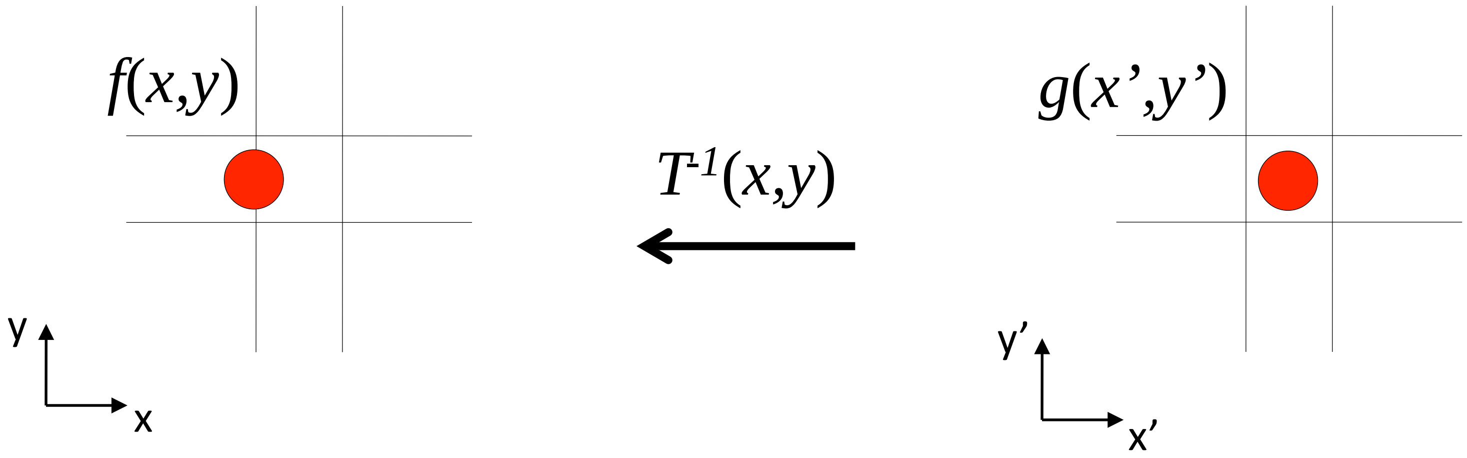


Inverse warping

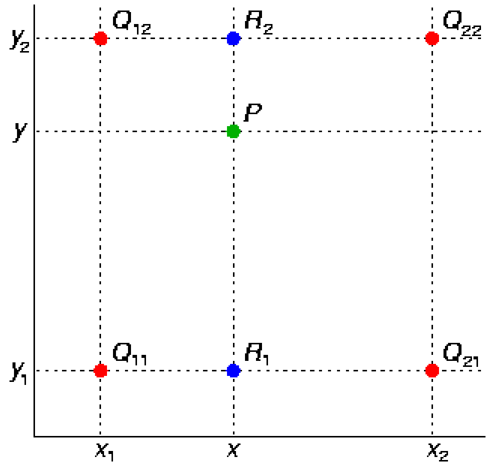
Pixel may come from between two points

- How do we determine its intensity?

✓ Use interpolation

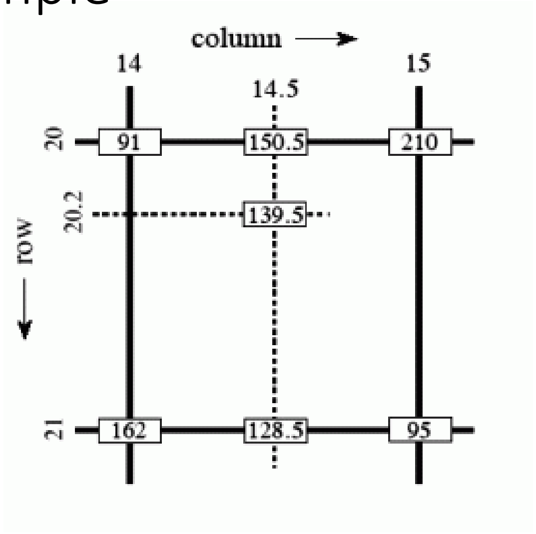


Bilinear interpolation




```
1.Interpolate to find R2
2.Interpolate to find R1
3.Interpolate to find P
```

Grayscale example



In matrix form (with adjusted coordinates)

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

Matlab:

call `interp2`

Python:

SciPy interp2d()

https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp2d.html

```
from scipy.interpolate import interp2d
f = interp2d(x, y, z, kind='cubic')
znew = f(xnew, ynew)
```

Forward vs inverse warping

Suppose we have two images.

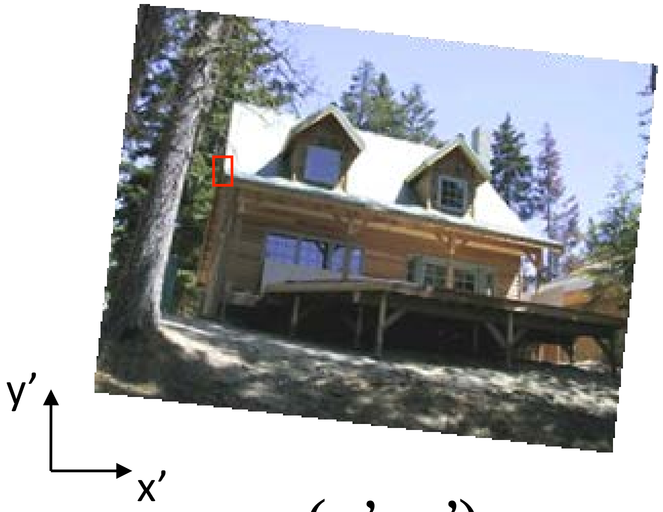
- How do we compute the transform that takes one to the other?

$f(x, y)$

$T(x, y)$

$\longleftrightarrow$

$T^{-1}(x, y)$



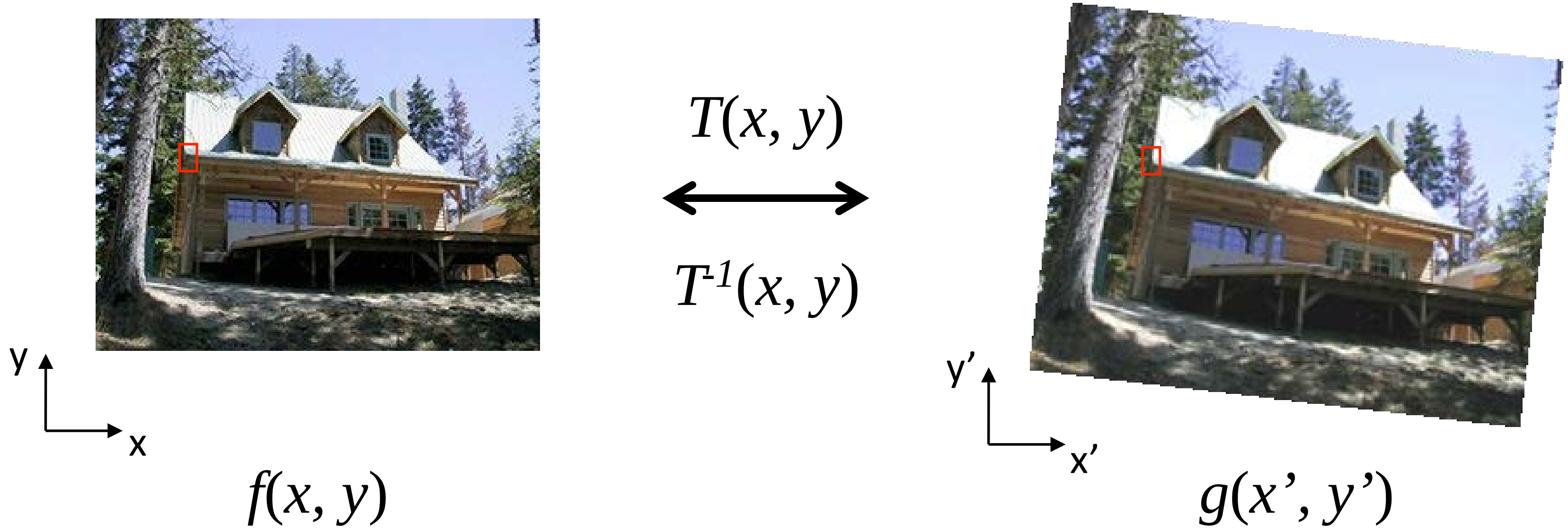
$g(x', y')$

Pros and cons of each?

Forward vs inverse warping

Suppose we have two images.

- How do we compute the transform that takes one to the other?



- Inverse warping eliminates holes in target image
- Forward warping does not require existence of inverse transform

References

Basic reading:

- Szeliski textbook, Section 3.6., 6.1

Additional reading:

- Hartley and Zisserman, “Multiple View Geometry in Computer Vision,” Cambridge University Press 2004.
  a comprehensive treatment of all aspects of projective geometry relating to computer vision, Sections 2 and 4 in particular discuss everything about homography estimation
- Richter-Gebert, “Perspectives on projective geometry,” Springer 2011.
  a beautiful, thorough, and very accessible mathematics textbook on projective geometry (available online for free from CMU’s library).