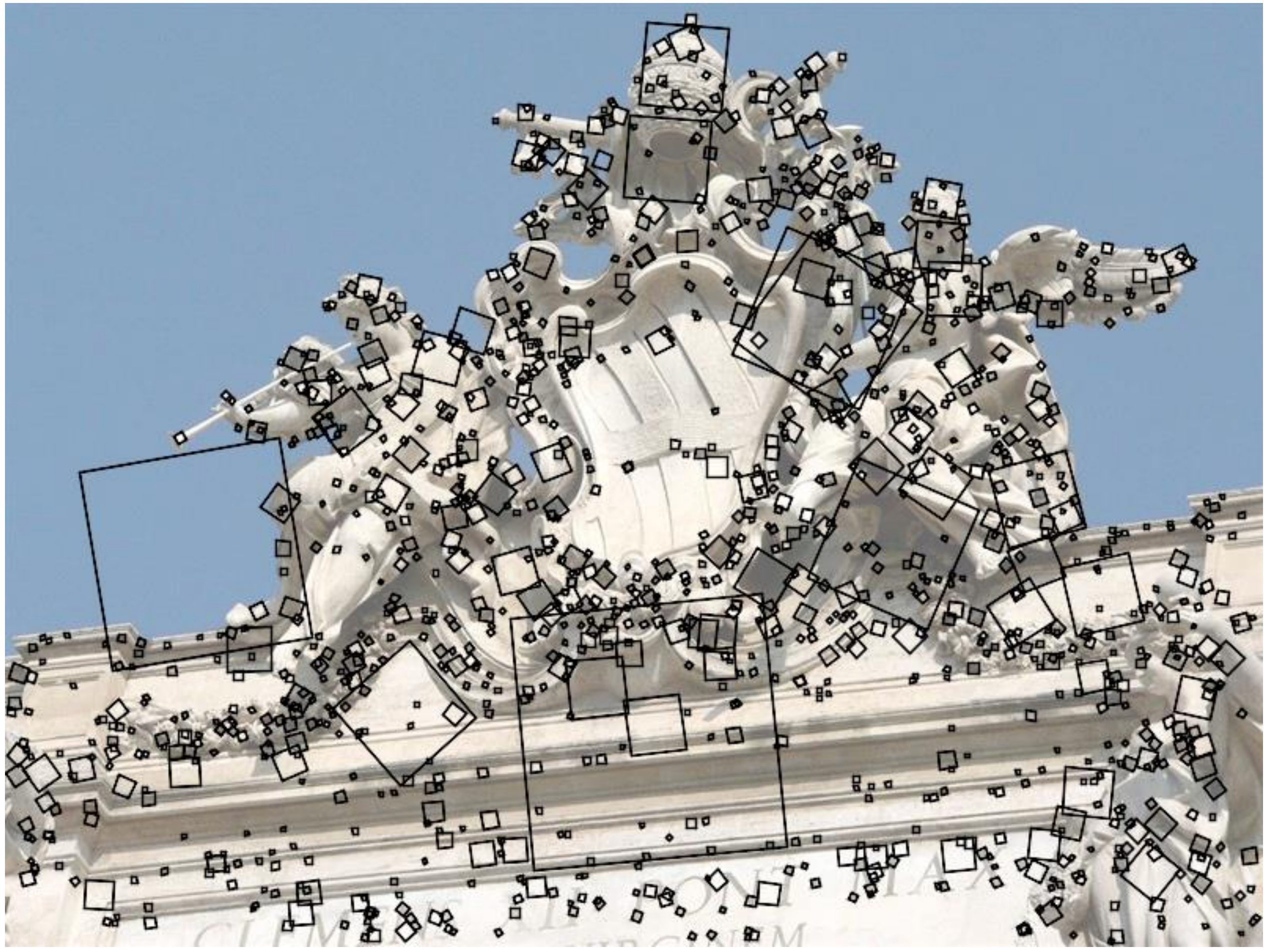
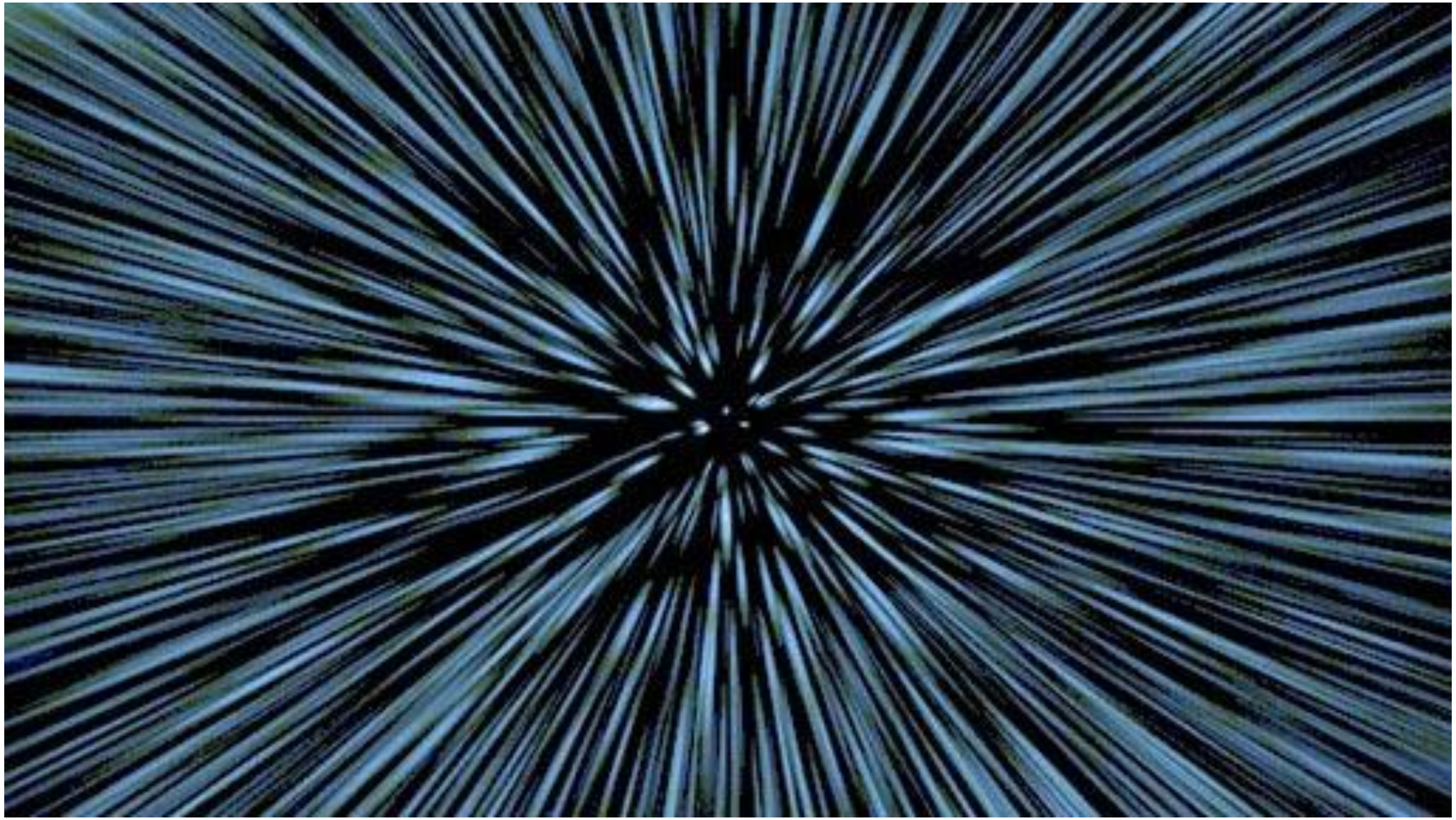
Correspondence



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



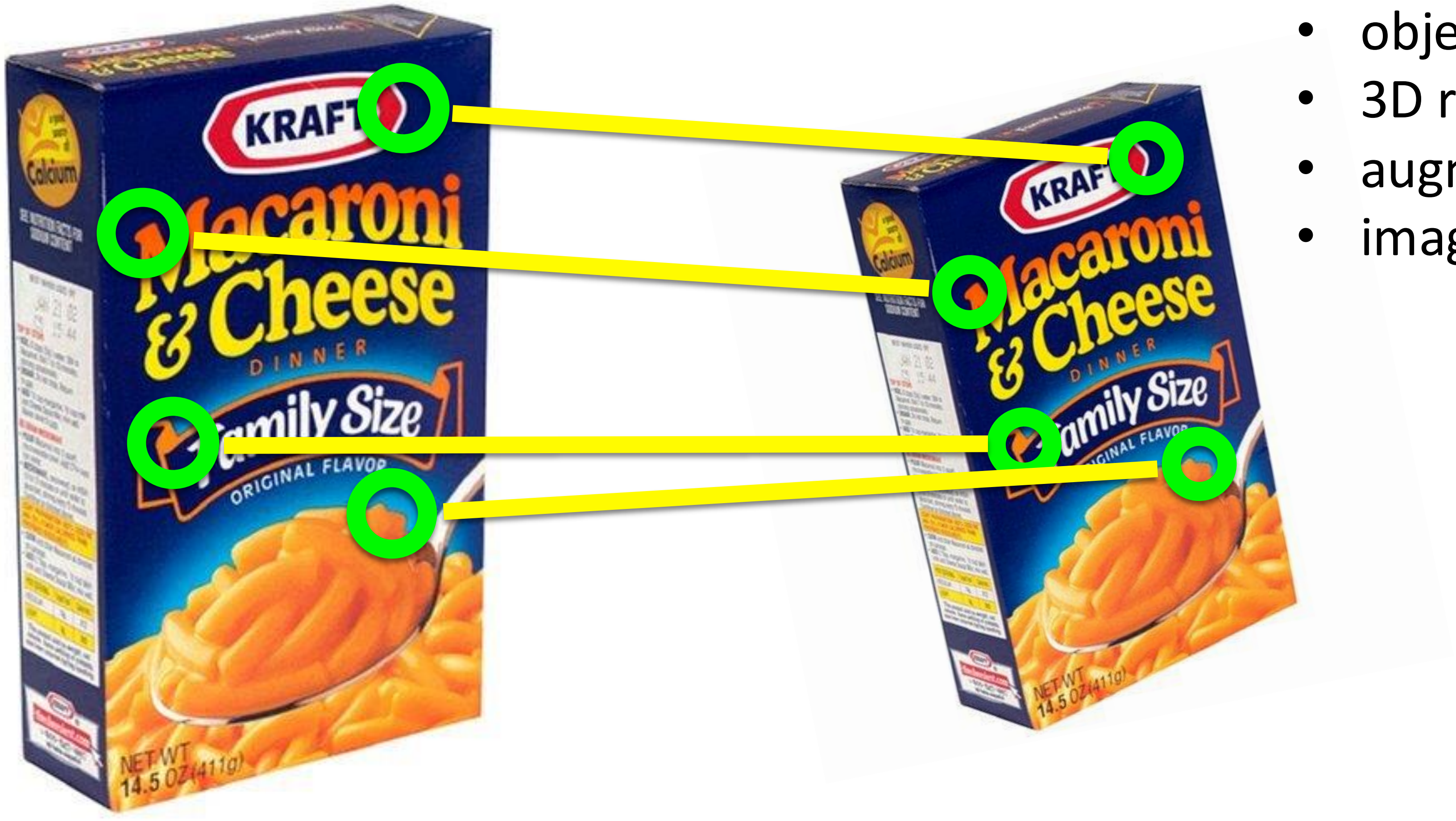
16-385 Computer Vision
Fall 2024, Lecture 7

http://16385.courses.cs.cmu.edu/

Overview of today’s lecture

- Reminder: image transformations.
- 2D transformations.
- Projective geometry 101.
- Transformations in projective geometry.
- Classification of 2D transformations.
- Determining unknown 2D transformations.
- Determining unknown image warps.

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:

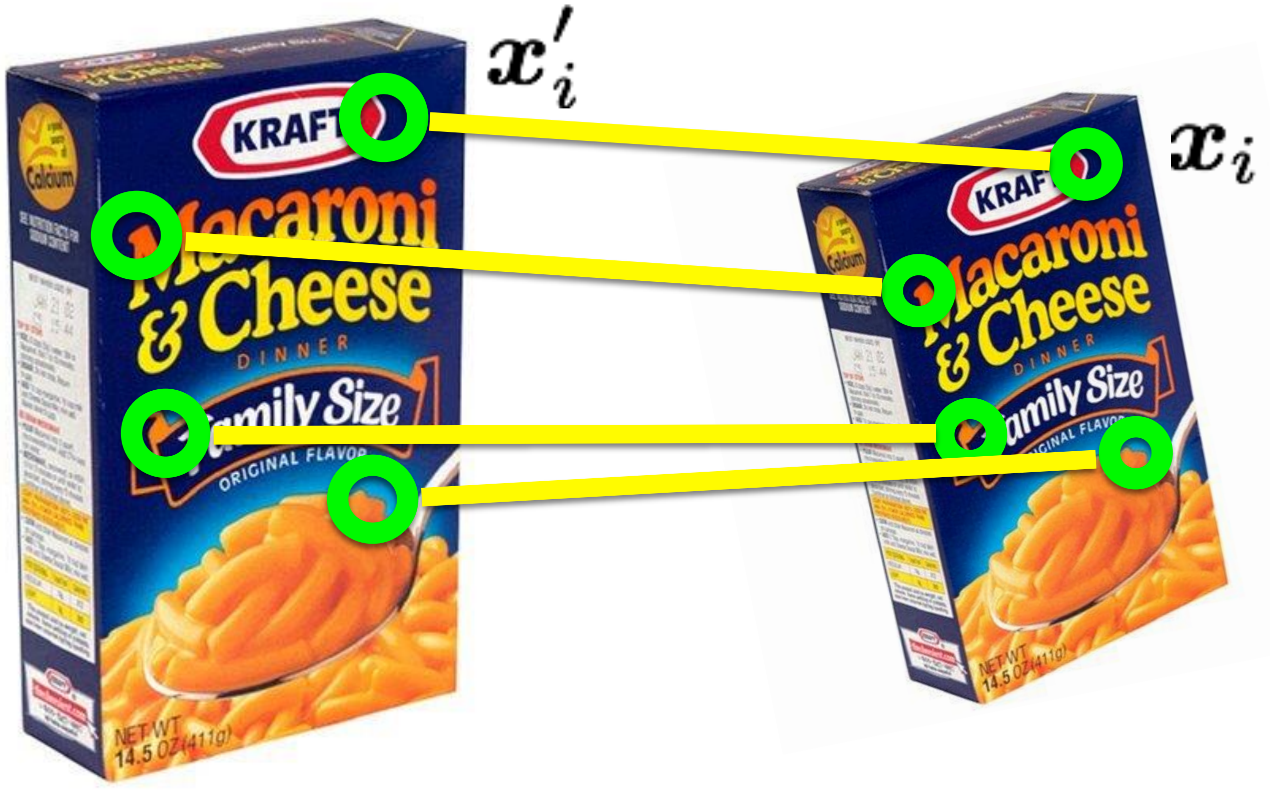
$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$

transformation function

parameters

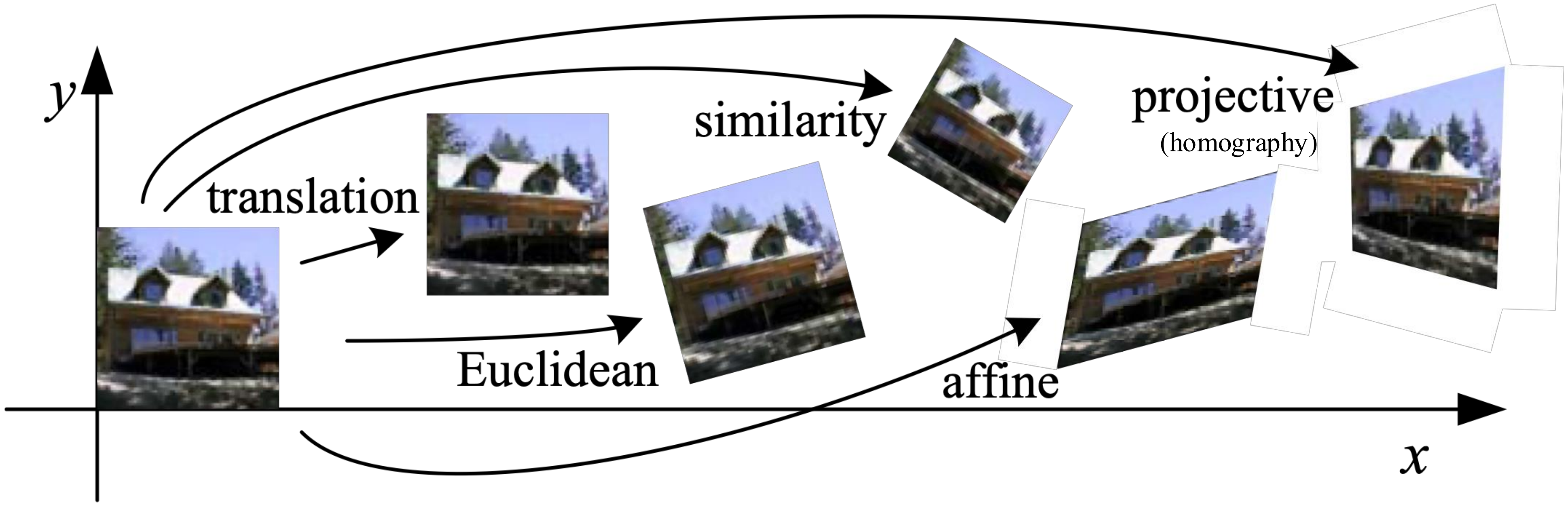
find the best estimate of the parameters

$$\boldsymbol{p}$$



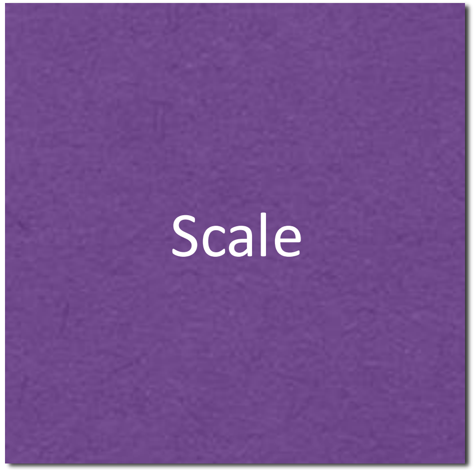
What kind of transformation functions $\boldsymbol{f}$ are there?

Family of image warps



2D planar transformations

2D planar transformations

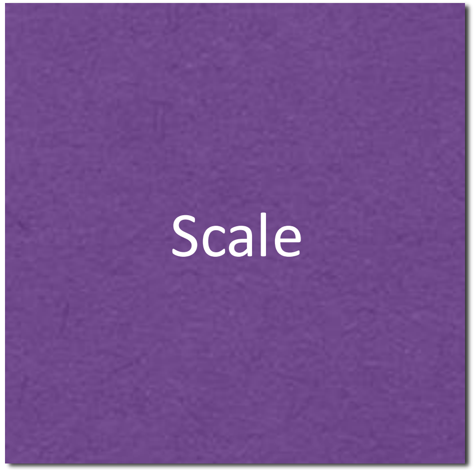


How would you implement scaling?

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

2D planar transformations

y



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

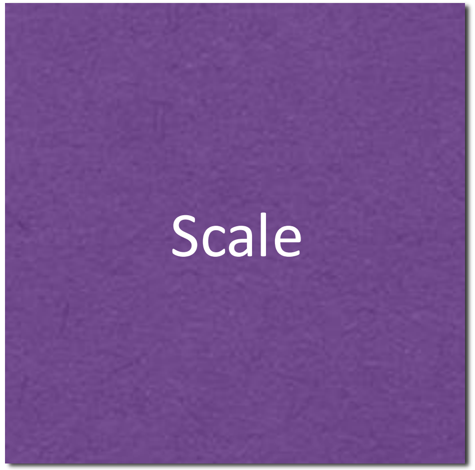
What's the effect of using different scale factors?

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

x

2D planar transformations

y



$$x' = ax$$

$$y' = by$$

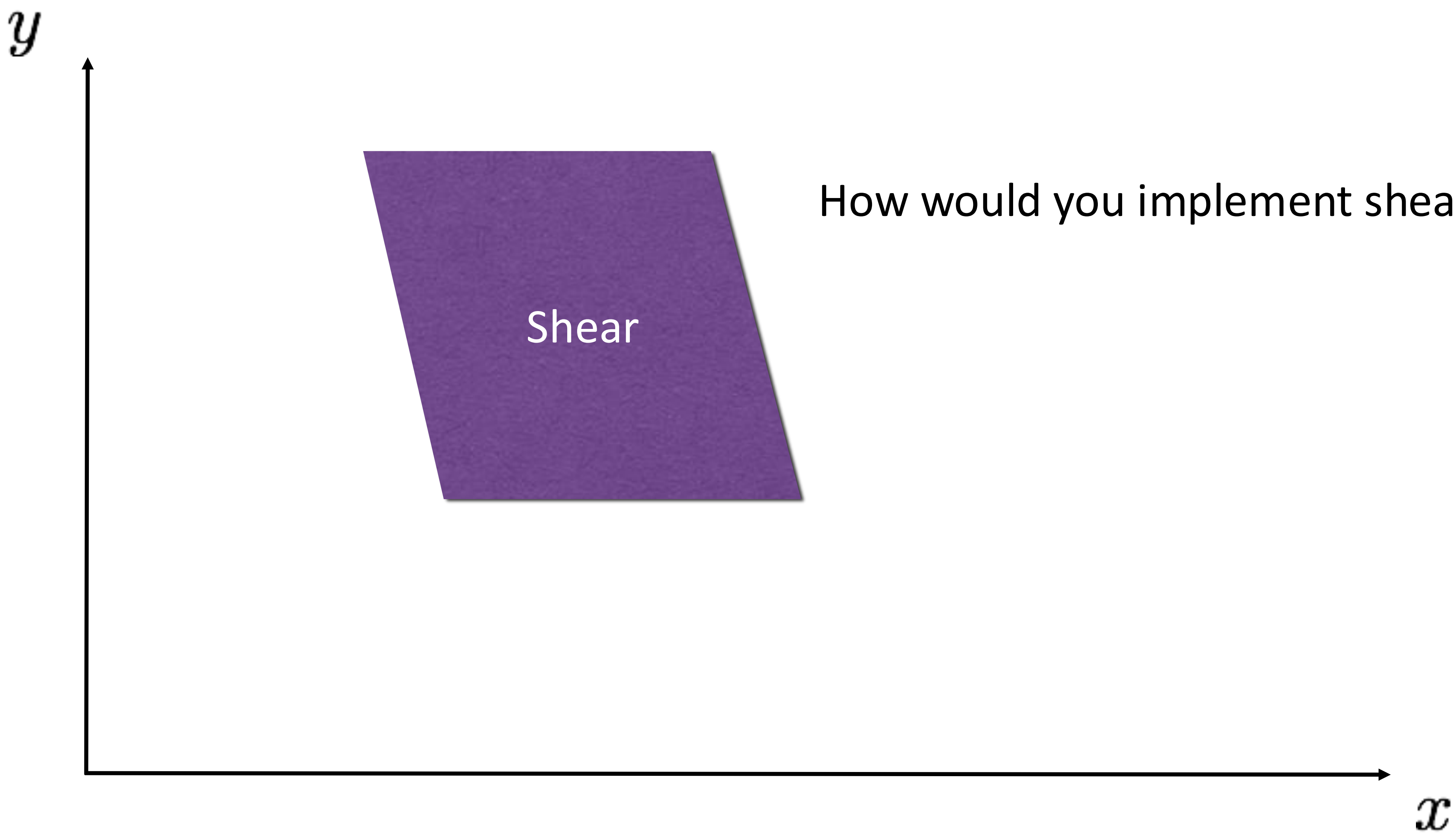
matrix representation of scaling:

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

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

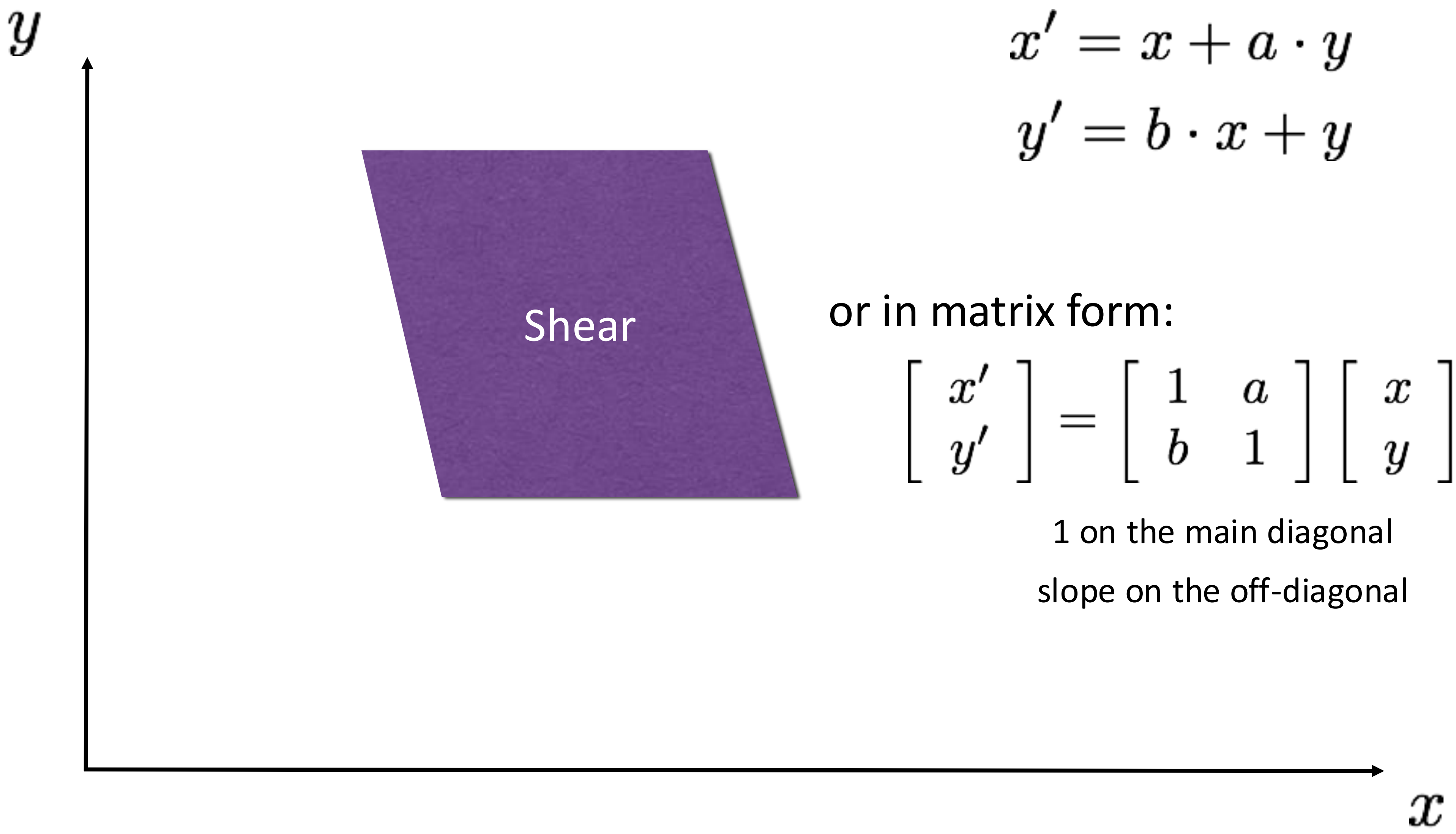
x

2D planar transformations



How would you implement shearing?

2D planar transformations



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

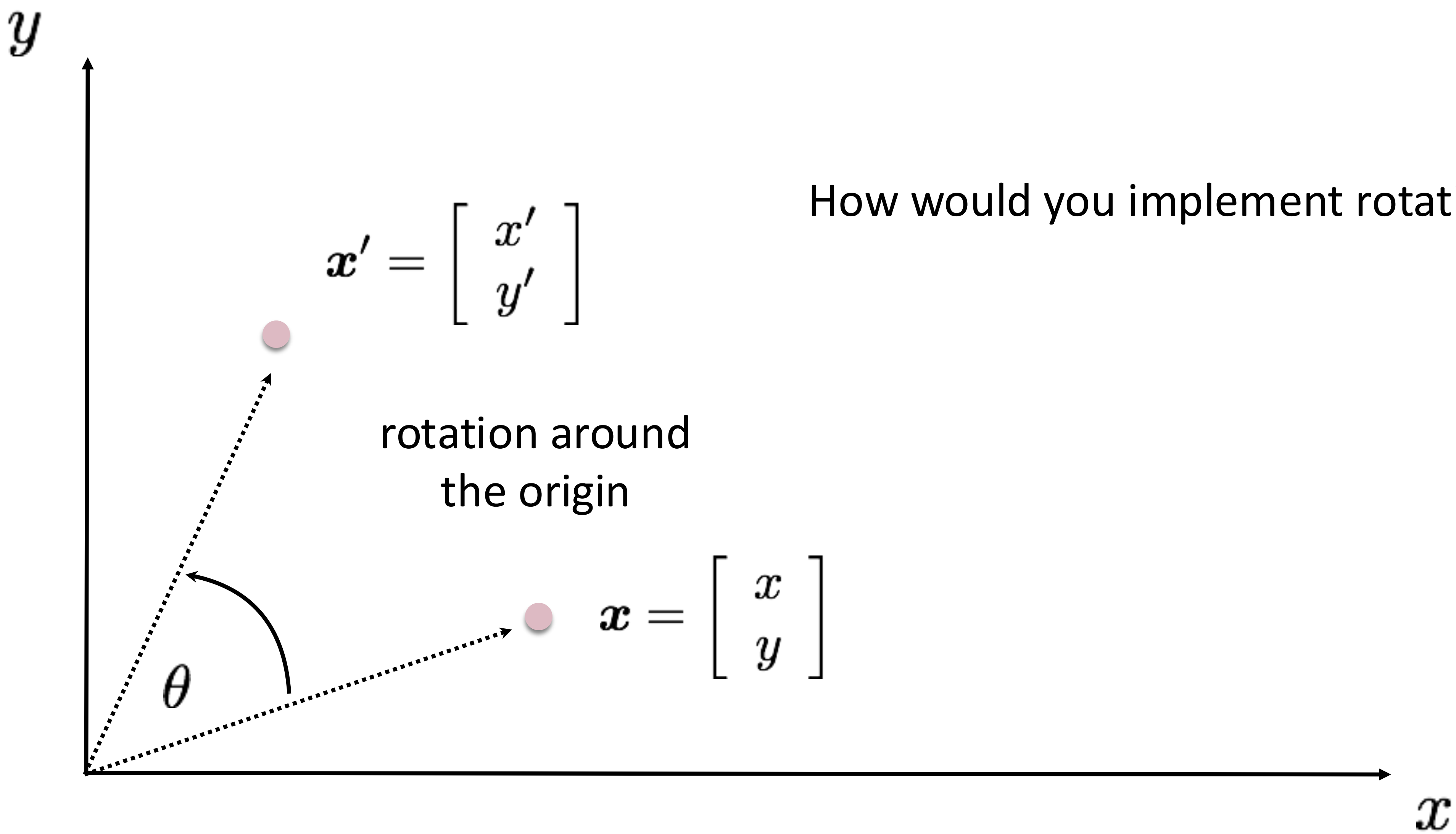
or in matrix form:

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

1 on the main diagonal

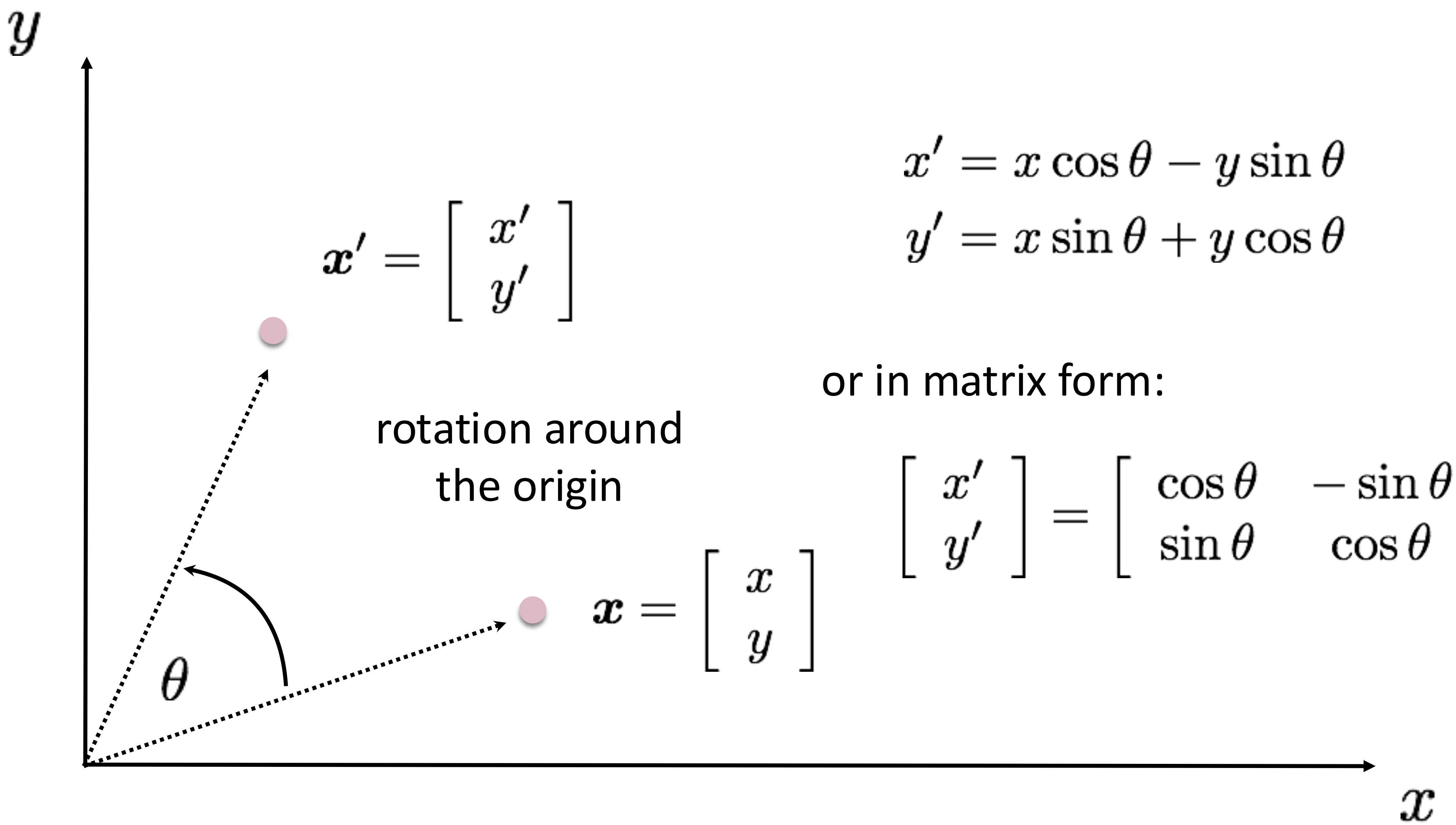
slope on the off-diagonal

2D planar transformations



How would you implement rotation?

2D planar transformations



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

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 transformations

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

$$\downarrow$$

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

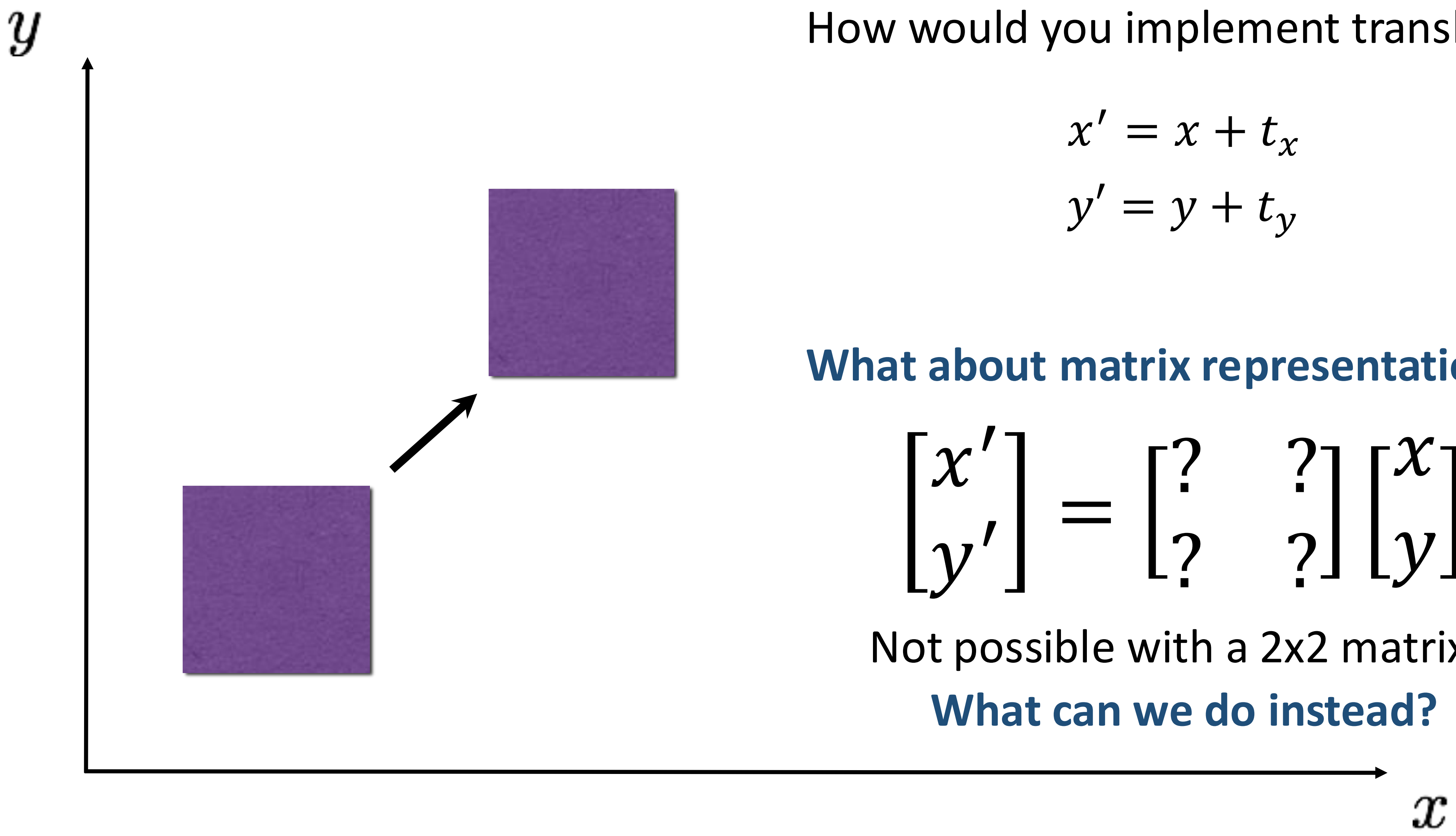
parameters p

point $\boldsymbol{x}$

Why do we like using a matrix representation for a transformation?

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \boldsymbol{M_4 M_3 M_2 M_1} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D translation



How would you implement translation?

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

What about matrix representation?

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

Not possible with a 2x2 matrix!

What can we do instead?

Standard -> Homogeneous coordinates

Standard image coordinates   homogeneous coordinates

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

add a 1 here

- Represent 2D point with 3D dimensions

Standard -> Homogeneous coordinates

Standard image 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 3D dimensions
- 3D vectors are only defined up to scale

Standard -> Homogeneous coordinates

Standard image coordinates

homogeneous coordinates

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

Standard -> Homogeneous coordinates

Standard image coordinates | homogeneous coordinates

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

Standard image coordinates | homogeneous coordinates

$$\begin{bmatrix} ? \\ ? \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

How do we convert from homogenous back to standard coordinates?

Standard -> Homogeneous coordinates

Standard image coordinates | homogeneous coordinates

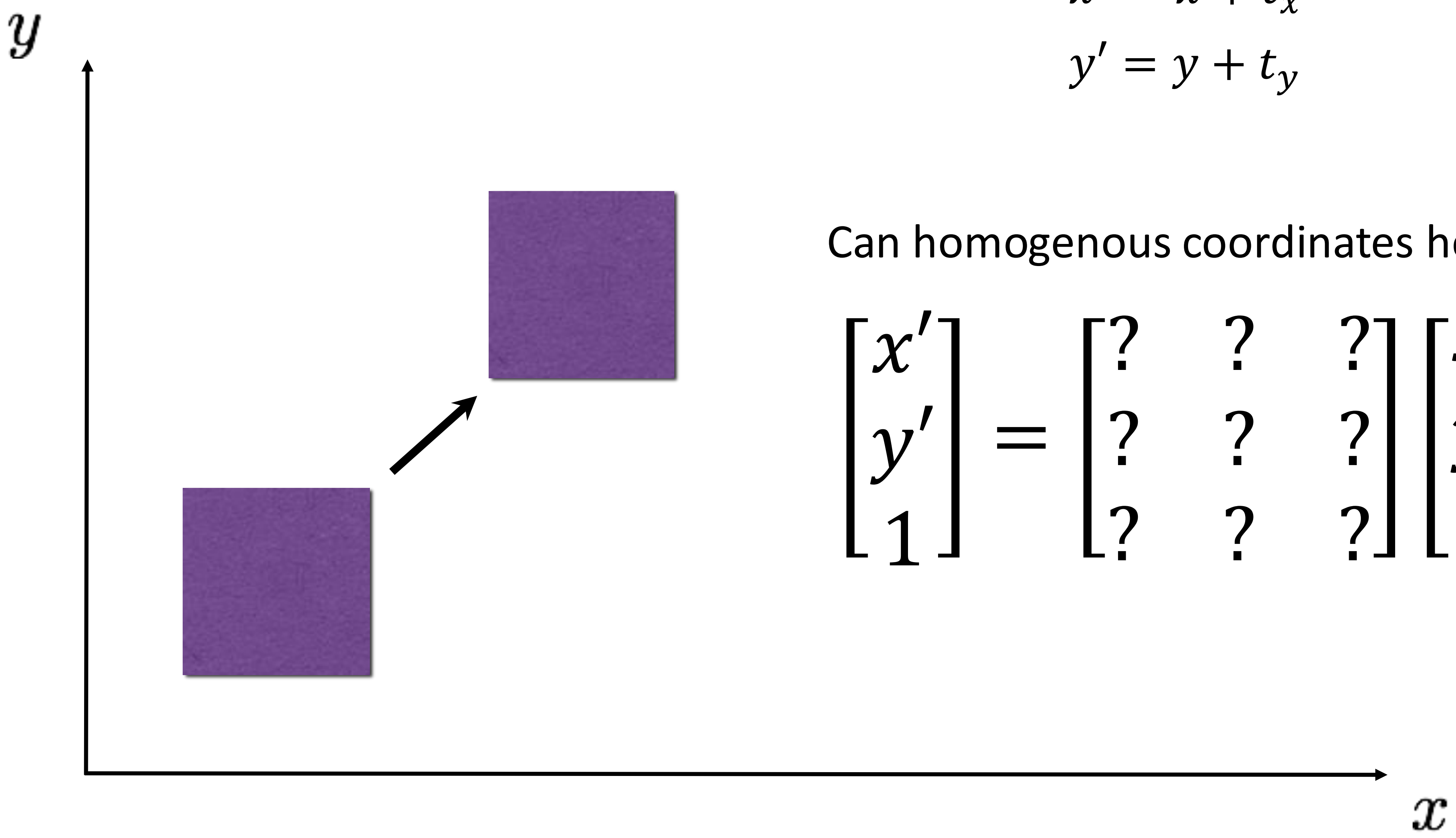
$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Standard image coordinates | homogeneous coordinates

$$\begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

How do we convert from homogenous back to standard coordinates?

2D translation

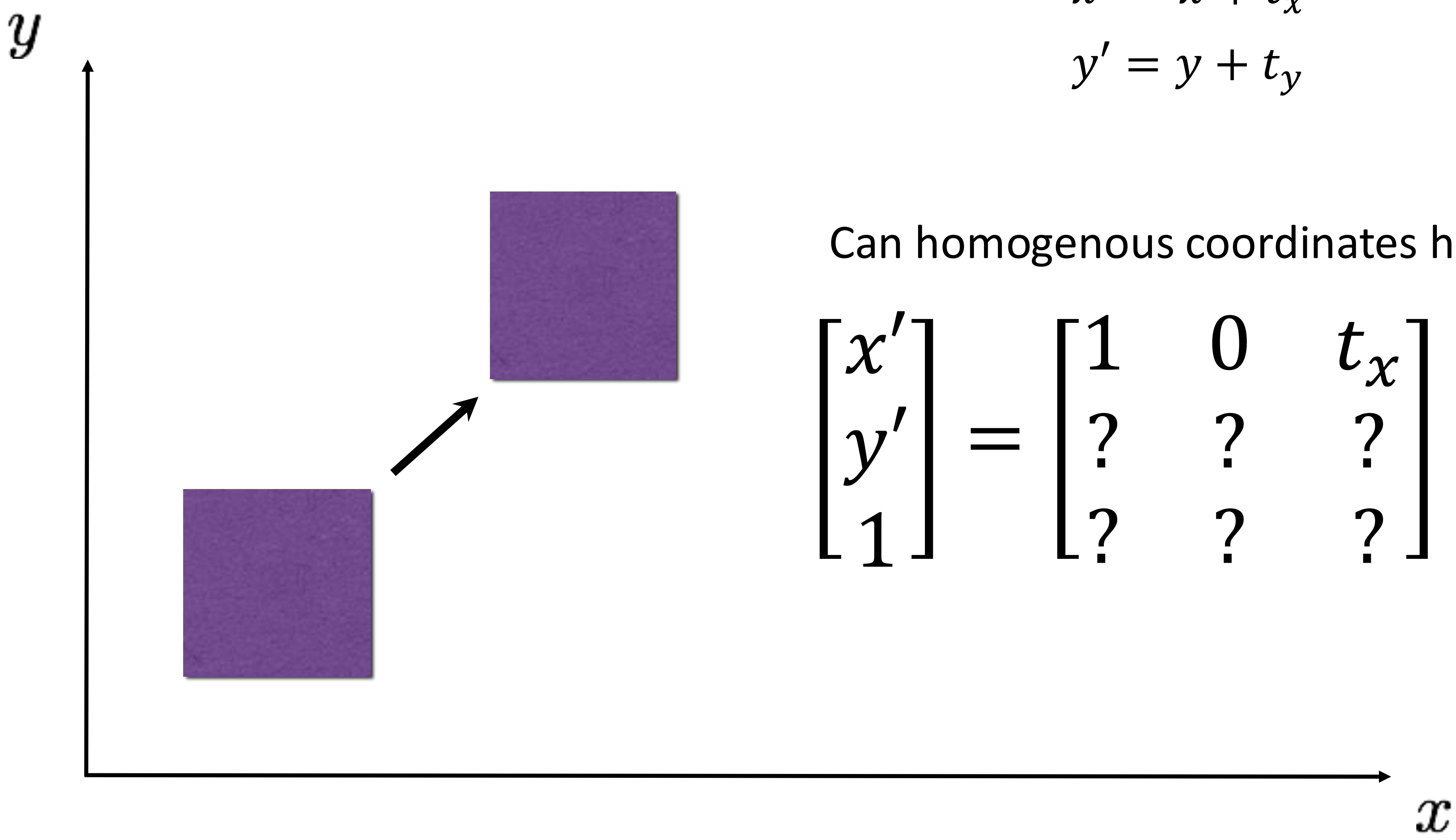


$$x' = x + t_x$$
$$y' = y + t_y$$

Can homogenous coordinates help us?

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

2D translation

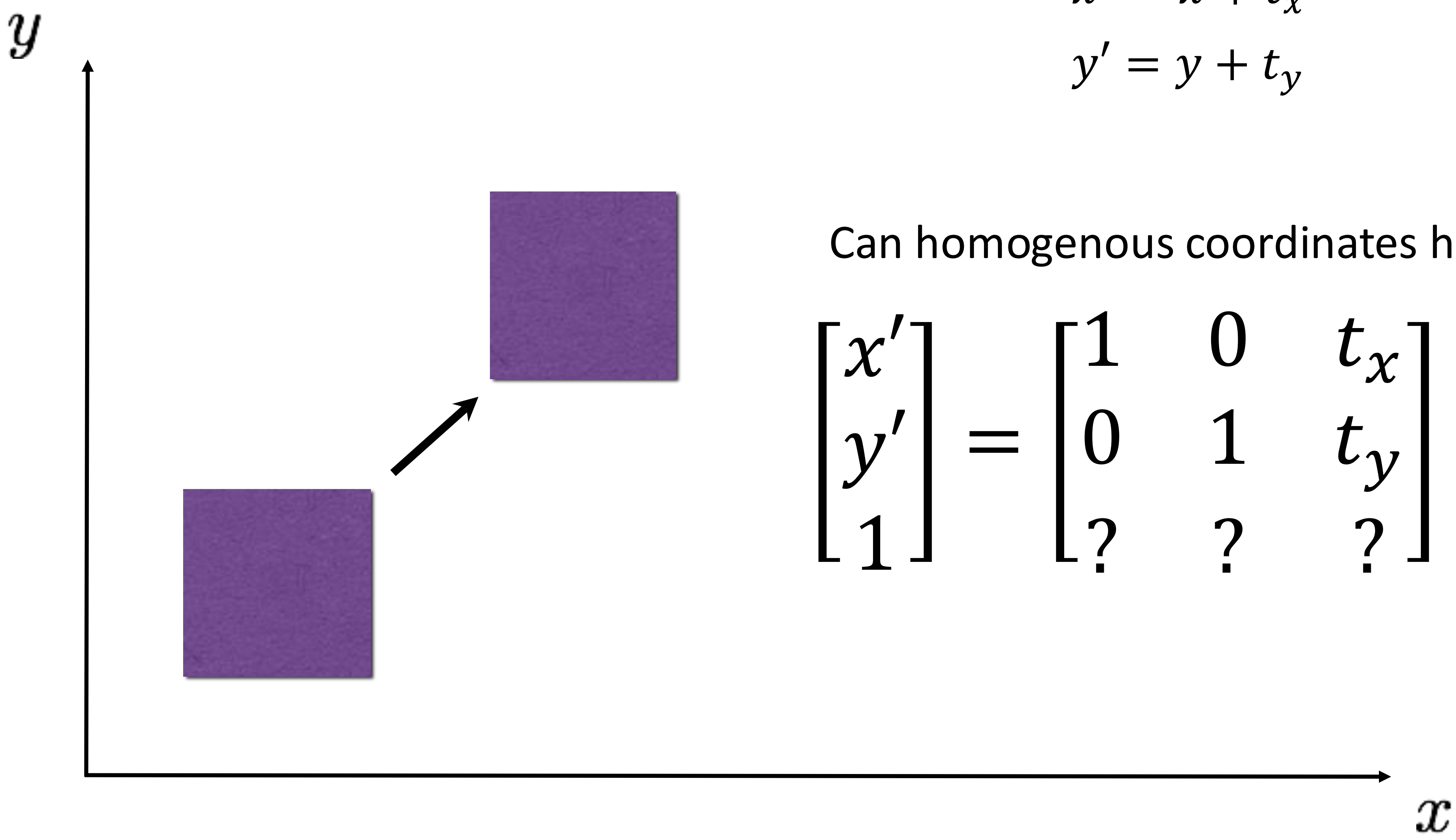


$$x' = x + t_x$$
$$y' = y + t_y$$

Can homogenous coordinates help us?

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

2D translation

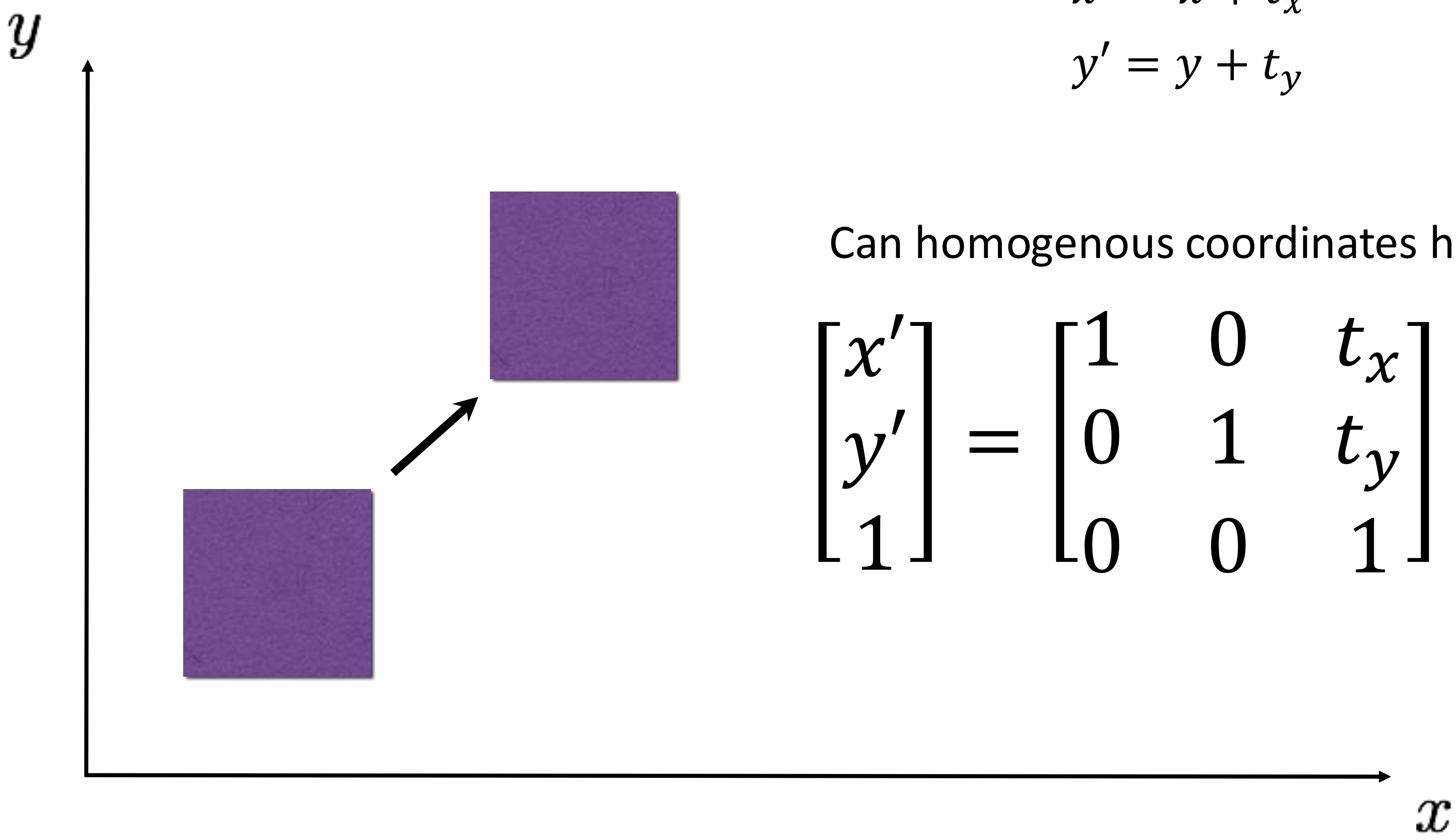


$$x' = x + t_x$$
$$y' = y + t_y$$

Can homogenous coordinates help us?

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

2D translation



$x' = x + t_x$

$y' = y + t_y$

Can homogenous coordinates help us?

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

Projective geometry

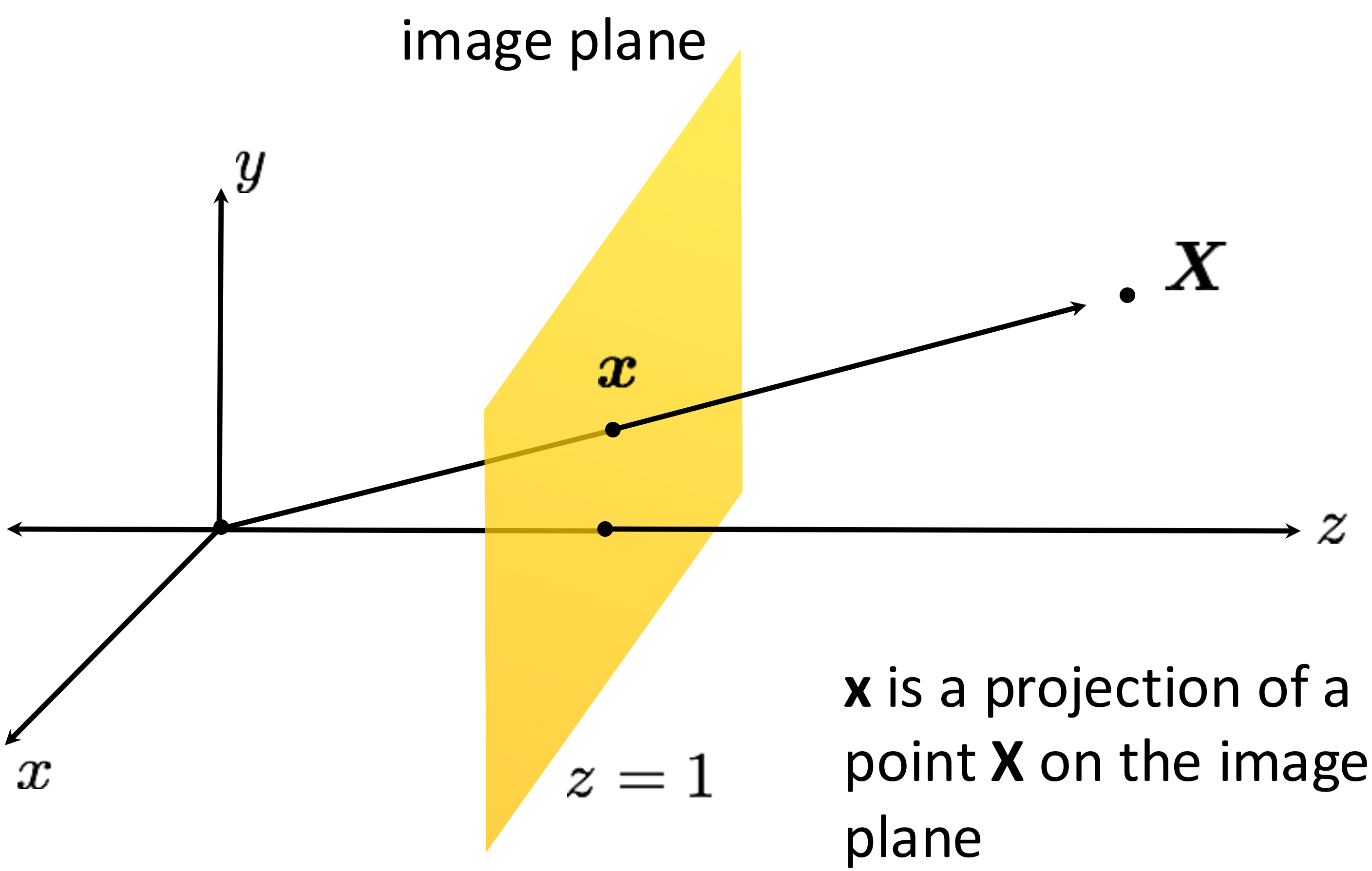
image point in standard (pixel) coordinates

$$\boldsymbol{x} = \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix}$$

$$\Updownarrow$$

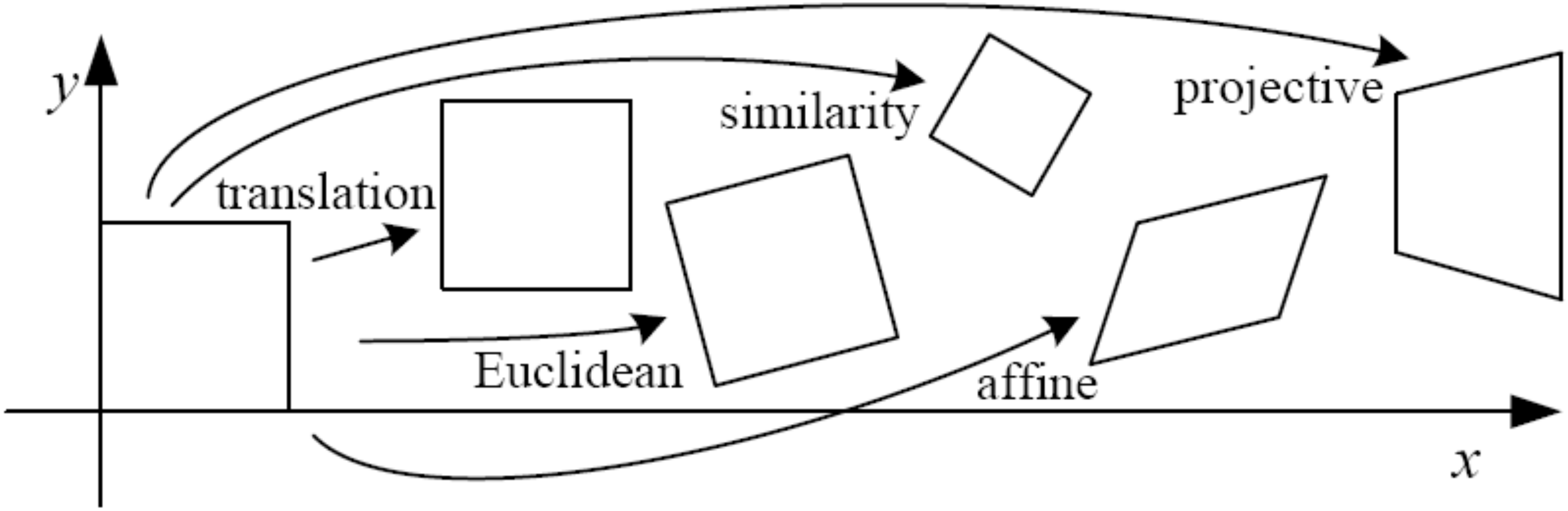
image point in homogeneous coordinates

$$\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



x is a projection of a point **X** on the image plane

Classification of 2D transformations

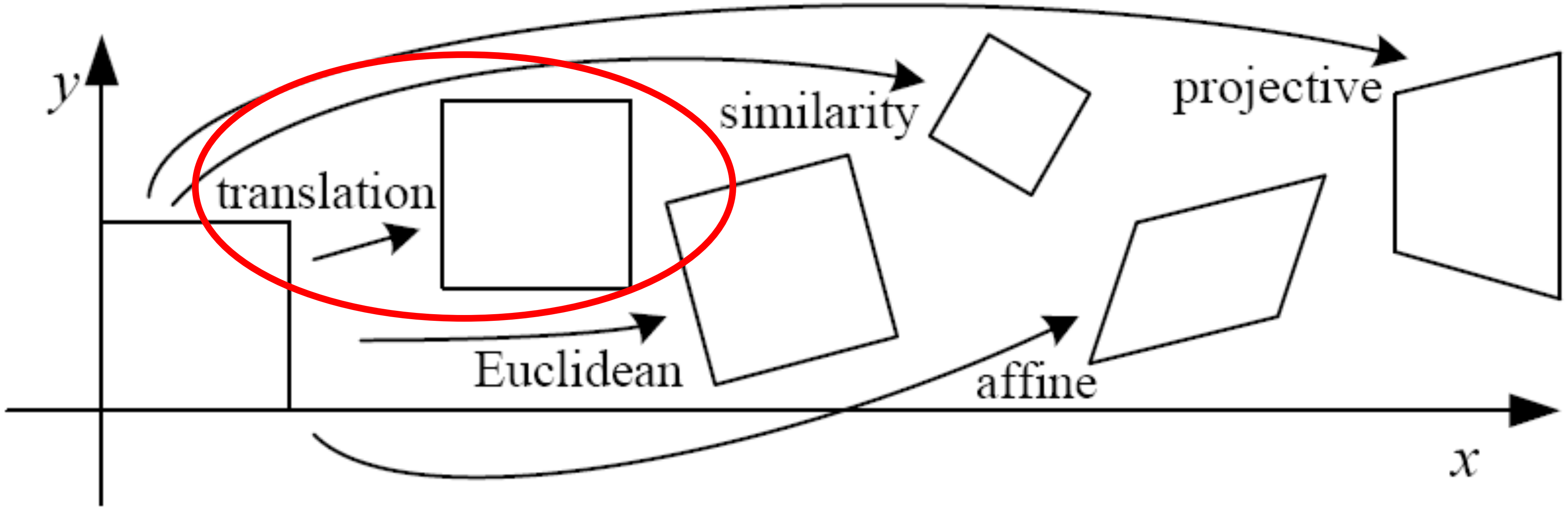


Classification of 2D transformations

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

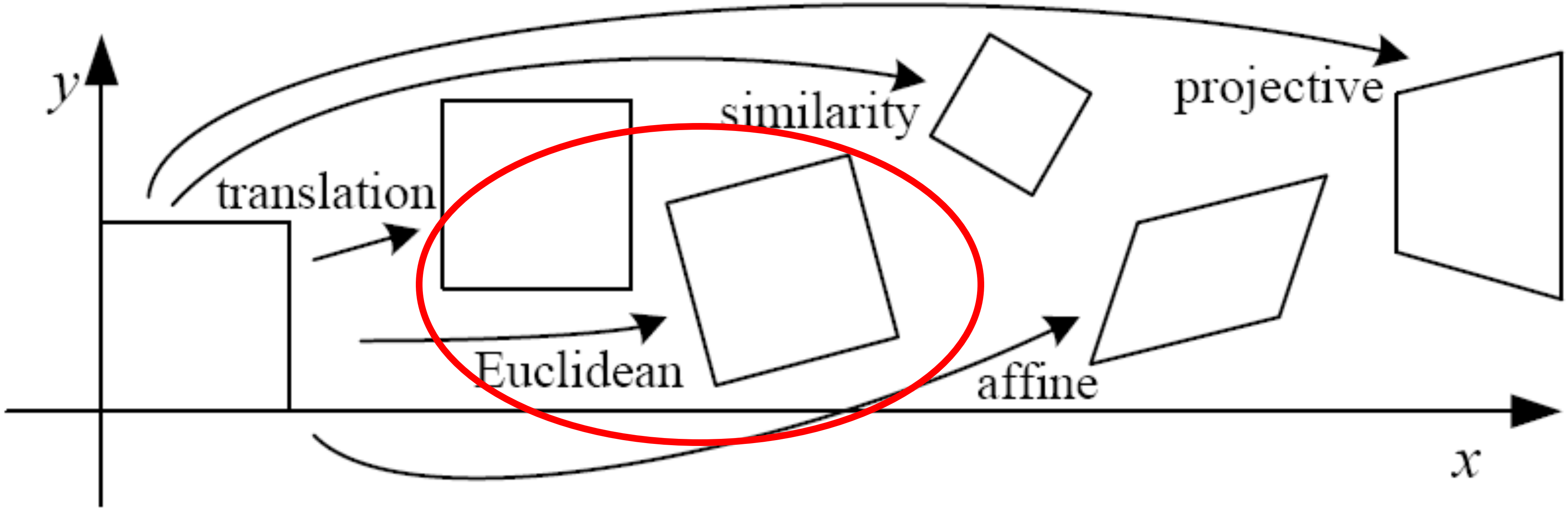
Euclidean (rigid):
rotation + translation

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

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

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

How many degrees of freedom?



Classification of 2D transformations

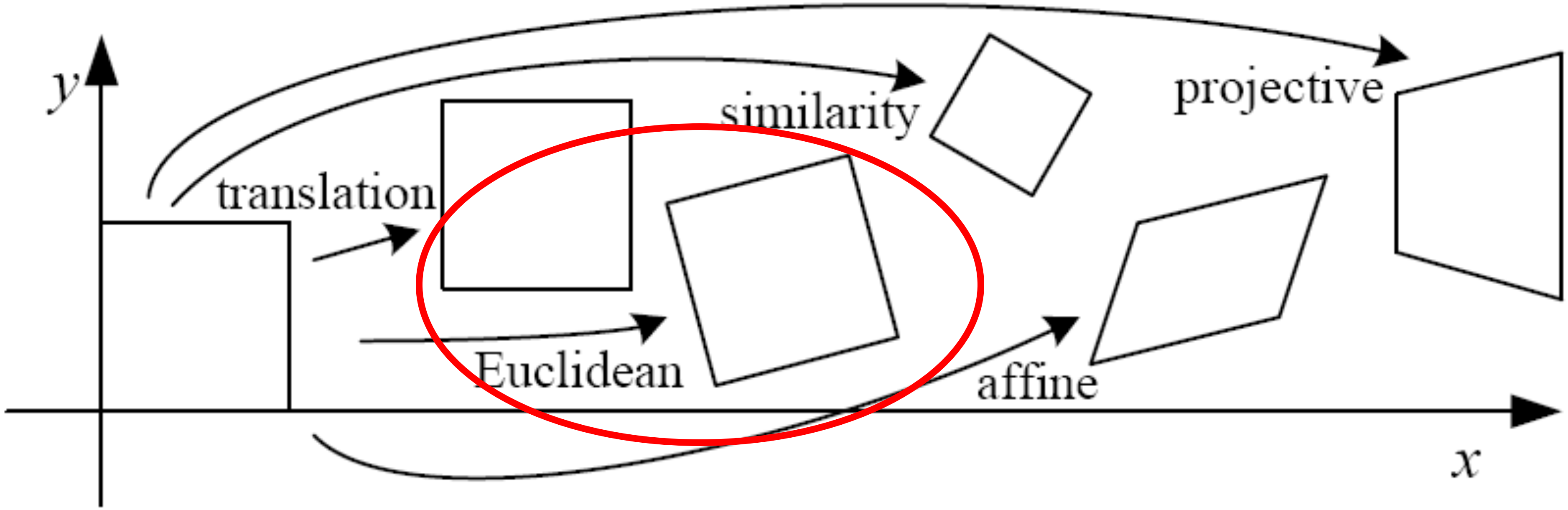
Euclidean (rigid):
rotation + translation

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

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

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

How many degrees of freedom?

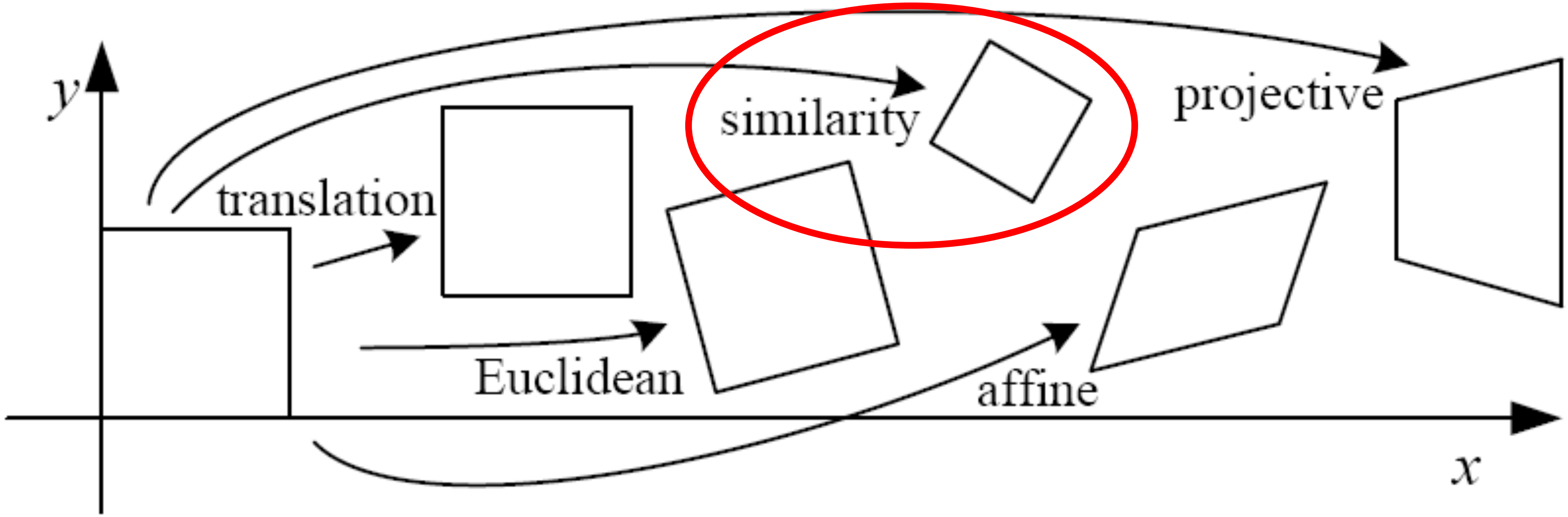


Classification of 2D transformations

Similarity:
uniform scaling + rotation
+ translation

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

How many degrees of freedom?



Classification of 2D transformations

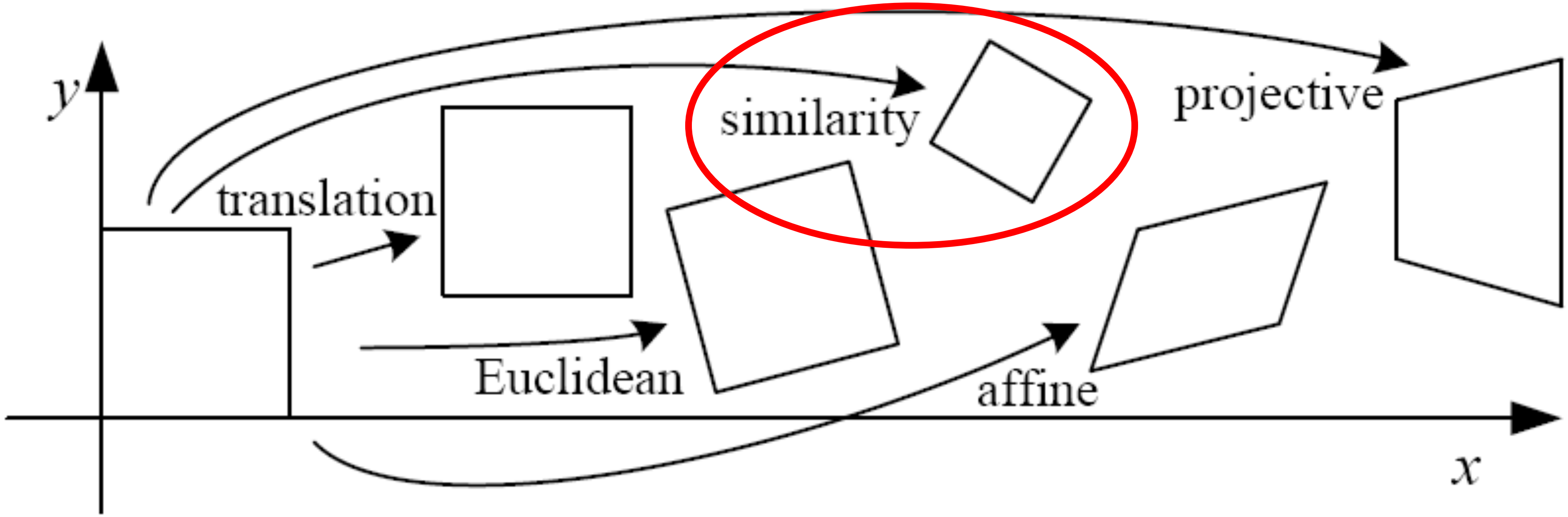
Similarity:
uniform scaling + rotation
+ translation

$$\begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

(this matrix assumes that we apply uniform scaling first, then translation / rotation)

Translation * Rotation * Scale * x

How many degrees of freedom?



Classification of 2D transformations

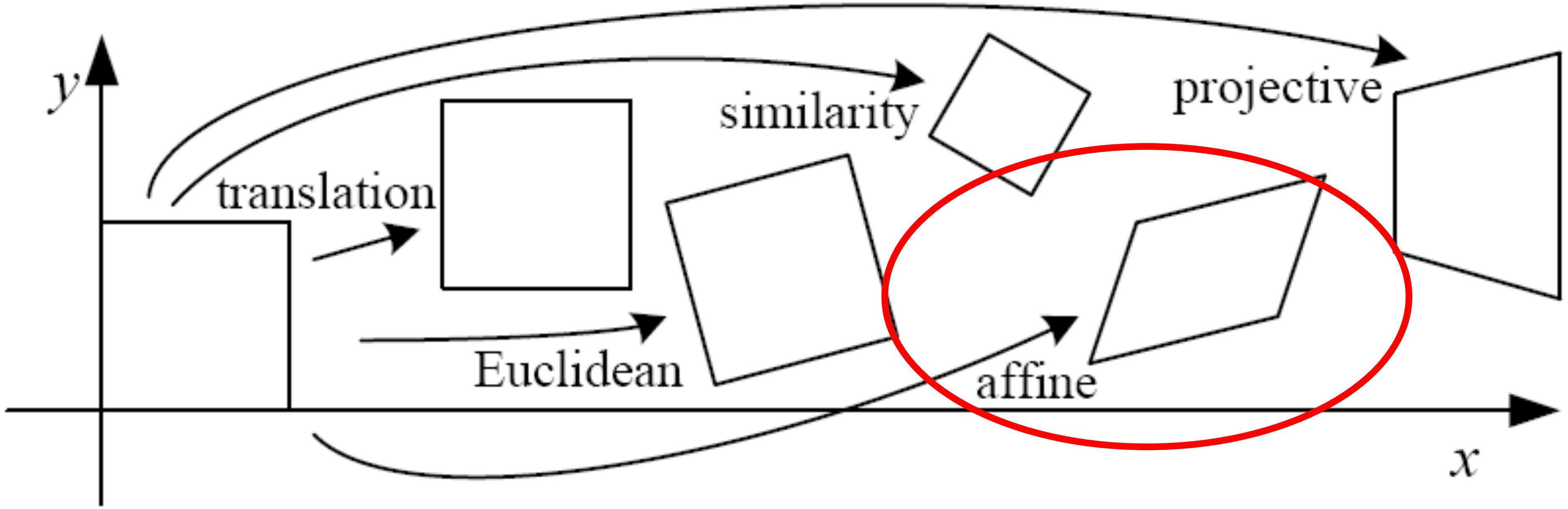
Affine transform:
uniform scaling + shearing
+ rotation + translation

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

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

Shear in x, y

How many degrees of freedom?



Classification of 2D transformations

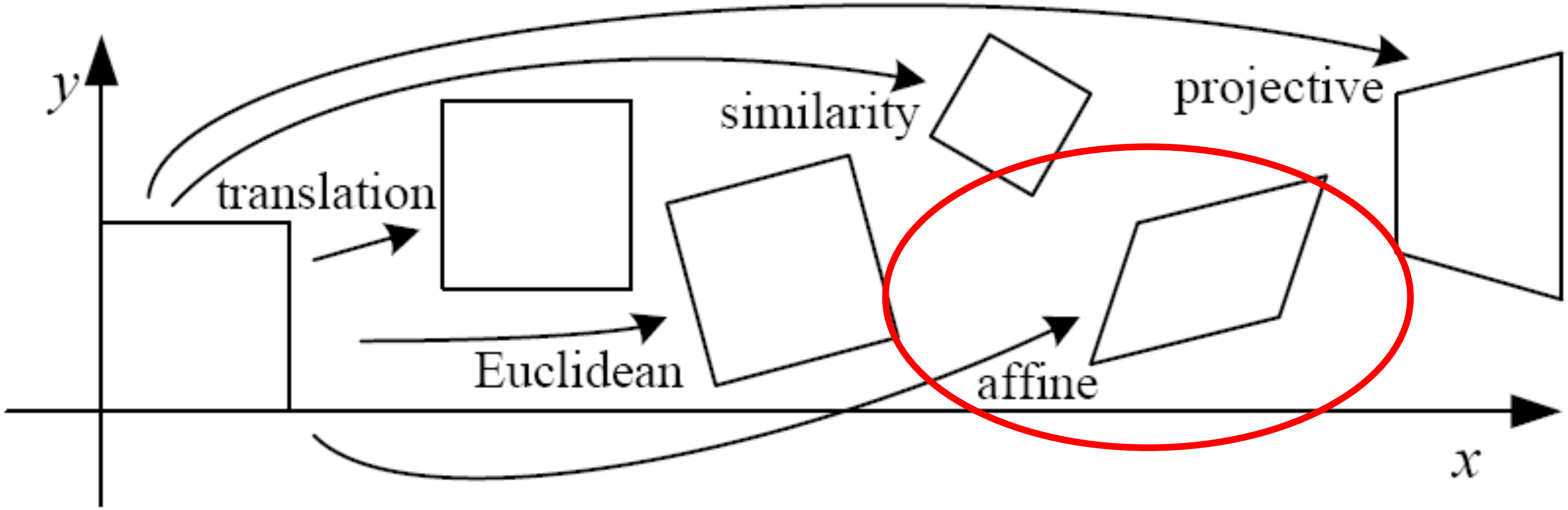
Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} s\cos\theta & -s\,\beta_x \sin\theta & t_x \\ s\,\beta_y \sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

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

Shear in x, y

How many degrees of freedom?



Affine transformations

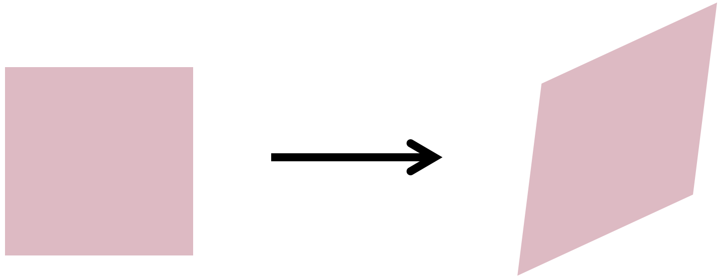
Affine transformations are combinations of

- 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:

- lines map to ?
- parallel lines map to ?
- ratios are ?
- compositions of affine transforms are ?



Affine transformations

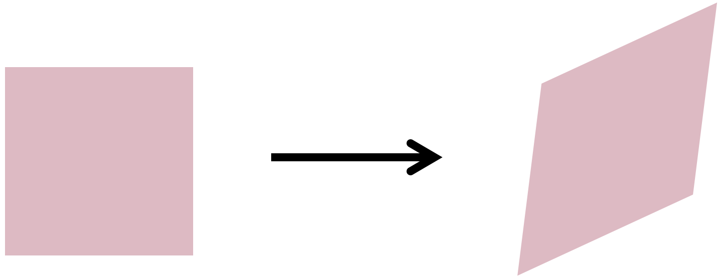
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Affine transformations are combinations of

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

Properties of affine transformations:

- lines map to lines
- parallel lines map to ?
- ratios are ?
- compositions of affine transforms are ?



Affine transformations

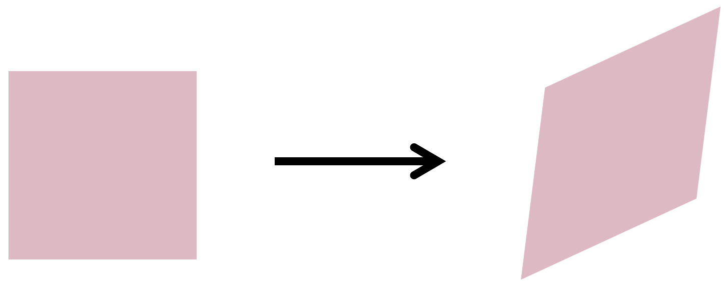
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Affine transformations are combinations of

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

Properties of affine transformations:

- lines map to lines
- parallel lines map to parallel lines
- ratios of segments within a line are ?
- compositions of affine transforms are ?



Affine transformations

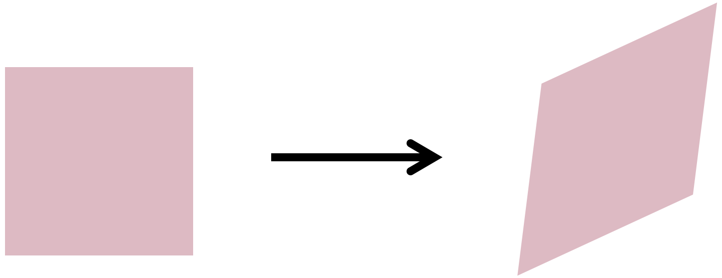
Affine transformations are combinations of

- 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:

- lines map to lines
- parallel lines map to parallel lines
- ratios of segments within a line are preserved
- compositions of affine transforms are ?



Affine transformations

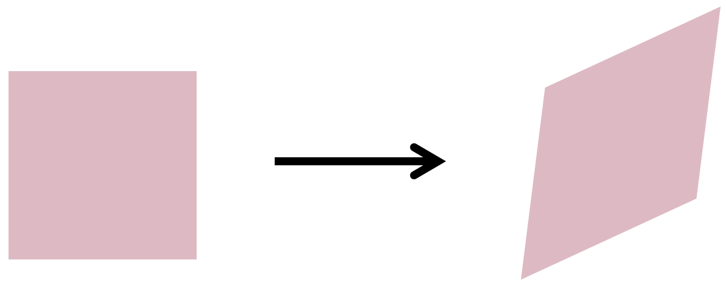
Affine transformations are combinations of

- 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:

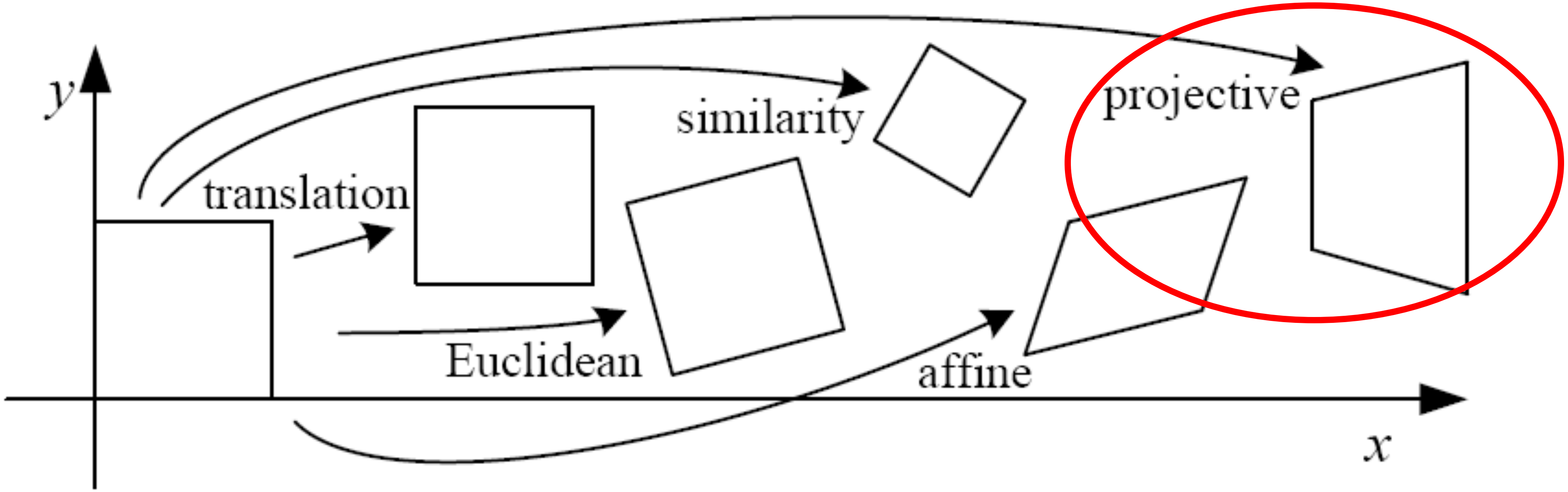
- lines map to lines
- parallel lines map to parallel lines
- ratios of segments within a line are preserved
- compositions of affine transforms are affine transforms



Projective transformations (aka homographies)

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

How many degrees of freedom?



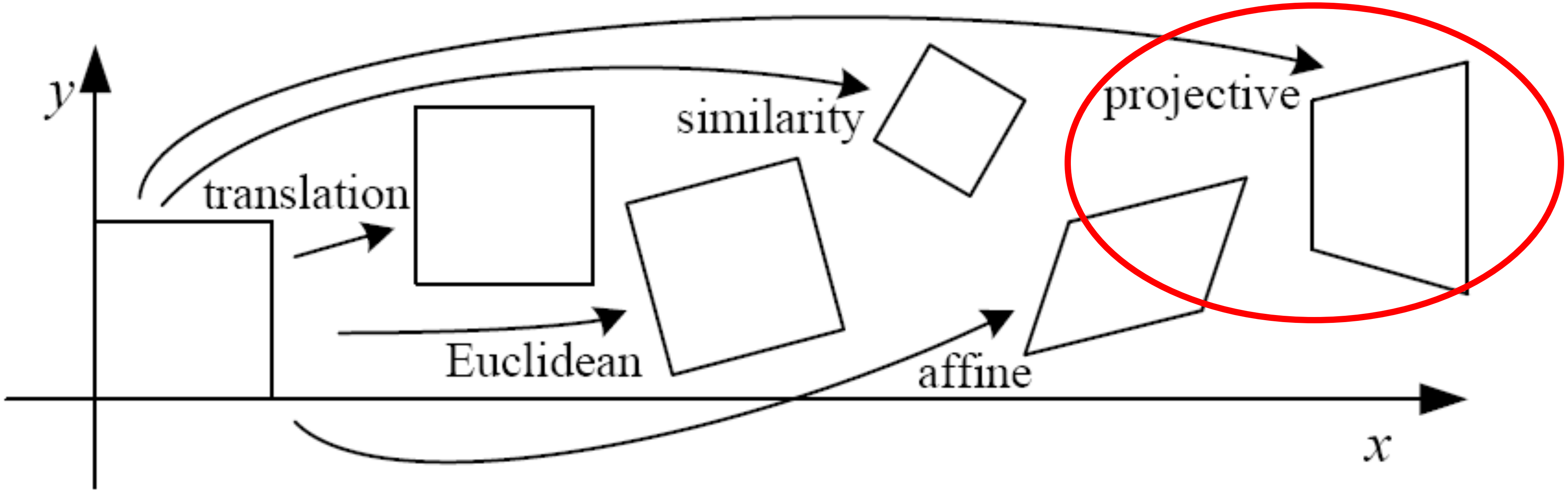
Projective transformations (aka homographies)

Properties of projective transformations:

- **Do lines map to lines?**
- **Do parallel lines map to parallel lines?**
- **Are ratios of segments within a line preserved?**
- **Are compositions of projective transforms are also projective transforms?**

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

8 DOF: vectors in homogenous coordinates are defined up to scale)



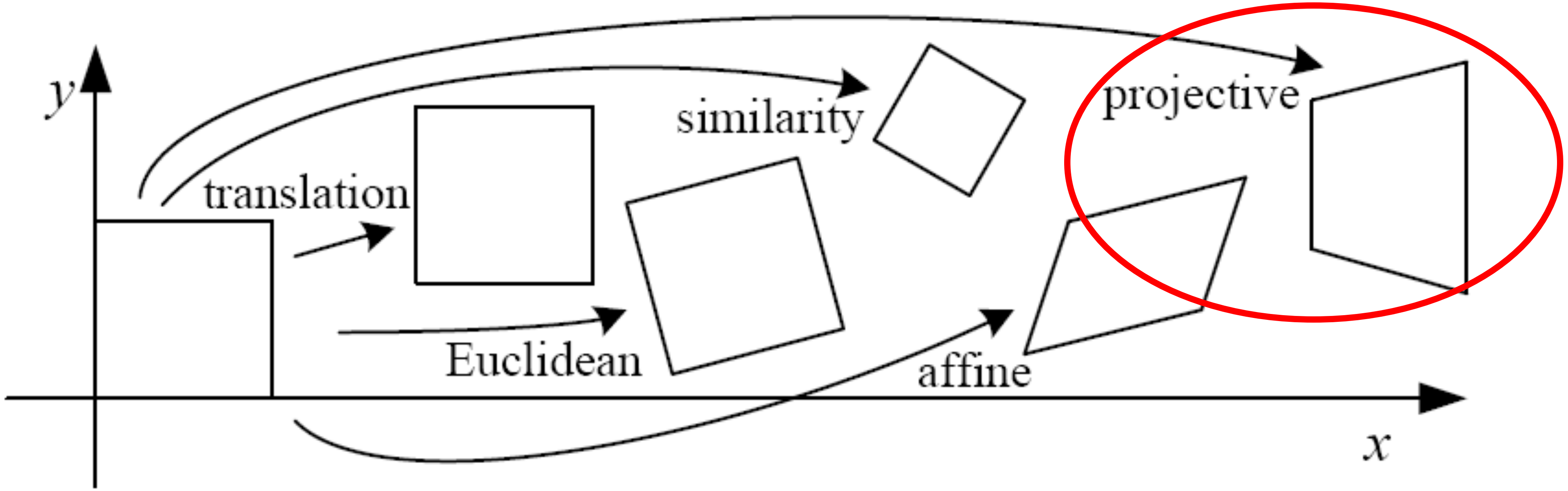
Projective transformations (aka homographies)

Properties of projective transformations:

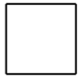
- Do lines map to lines? Yes
- Do parallel lines map to parallel lines? No
- Are ratios of segments within a line preserved? No
- Are compositions of projective transforms are also projective transforms? Yes

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

8 DOF: vectors in homogenous coordinates are defined up to scale)



Family of 2D warps

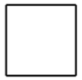
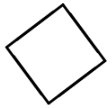
Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	

$$x' = x + t_x$$

$$y' = y + t_y$$

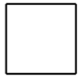
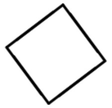
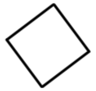
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	

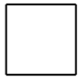
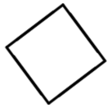
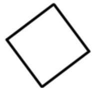
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} & \vert & \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} & \vert & \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} & \vert & \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles	

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

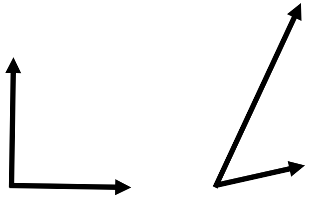
Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism	

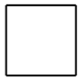
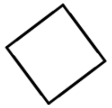
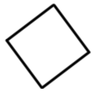
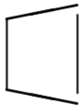
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

change of basis (rotate, scale x and y, rotate)

2D translation



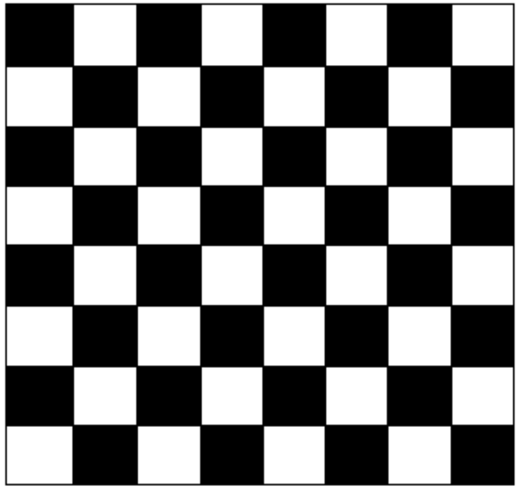
Family of 2D warps

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

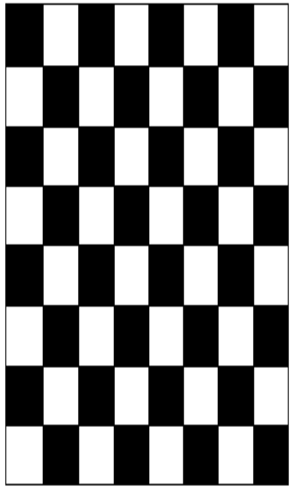
$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Relates the image projections of
(1) planar scene under any camera or (2) any scene under rotated cameras

Important property captured by 2D affine warps: *foreshortening*



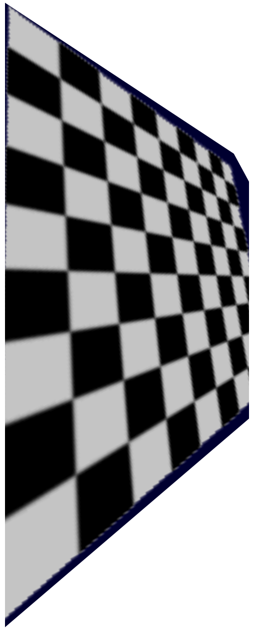
Fronto-parallel view



Affine warp

(Rotation of far-away plane)

All squares become
more narrow



Perspective projection

Homography warp

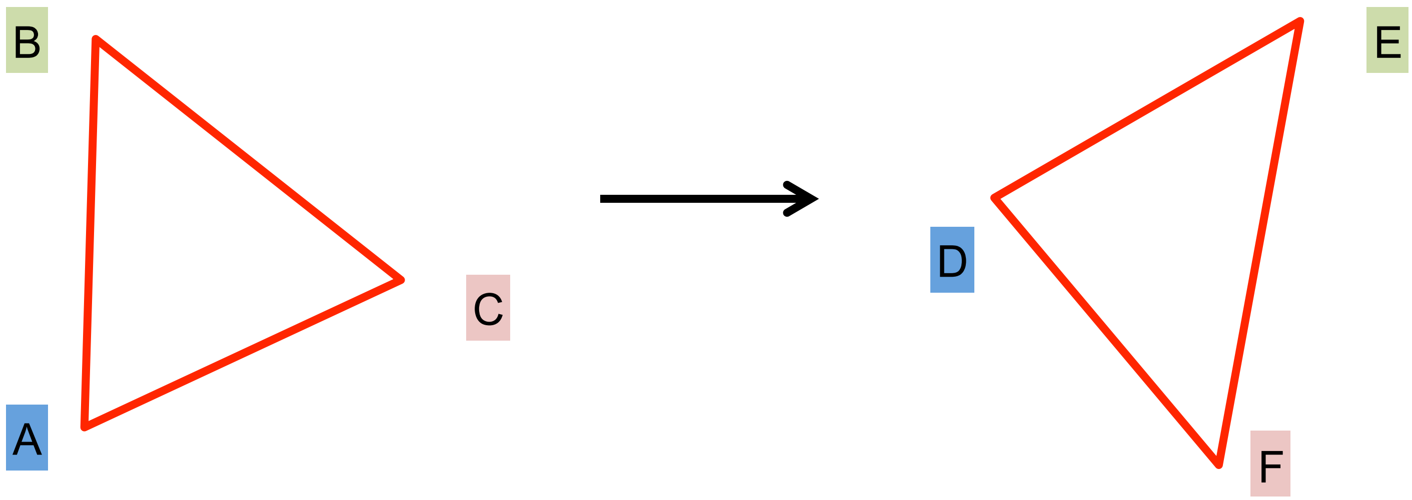
(Rotation of close-by plane)

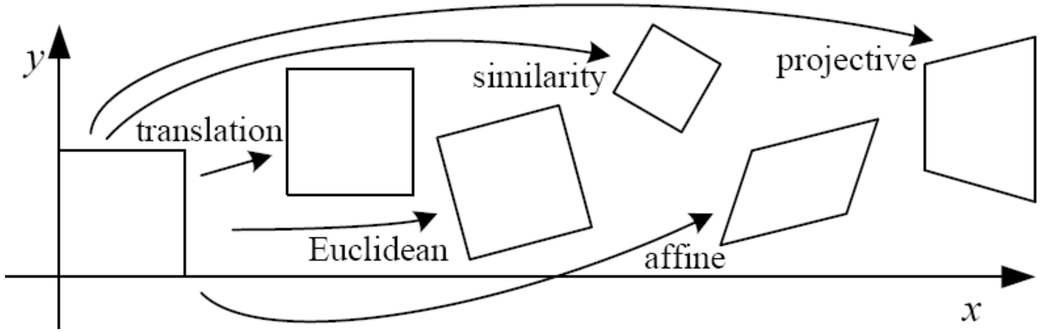
Far squares -> smaller
Close squares -> larger

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 many degrees of freedom do we have? 6 = 3 (x,y) coordinates

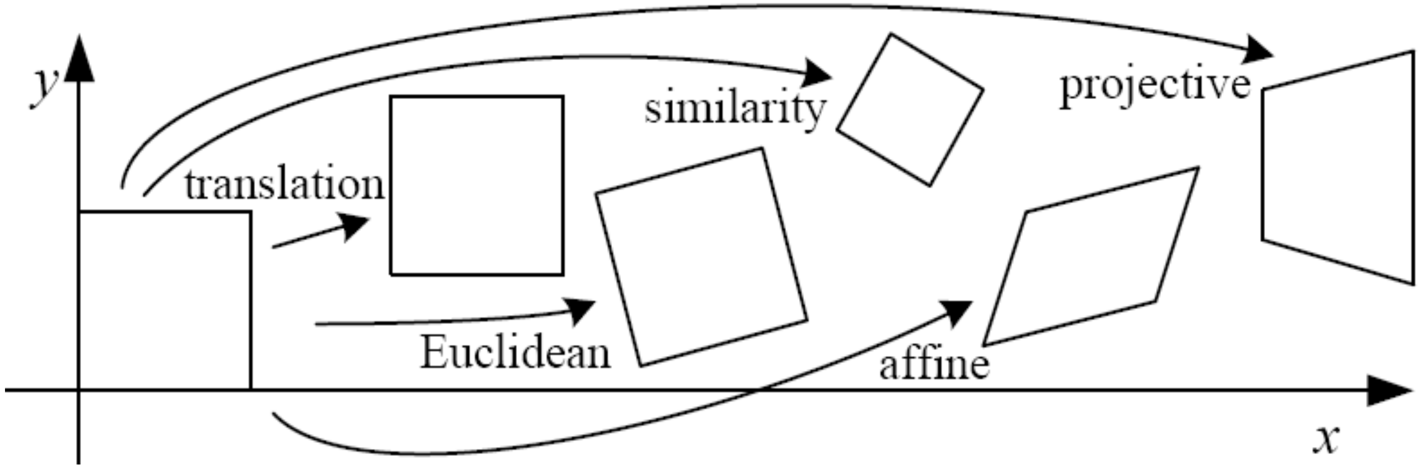


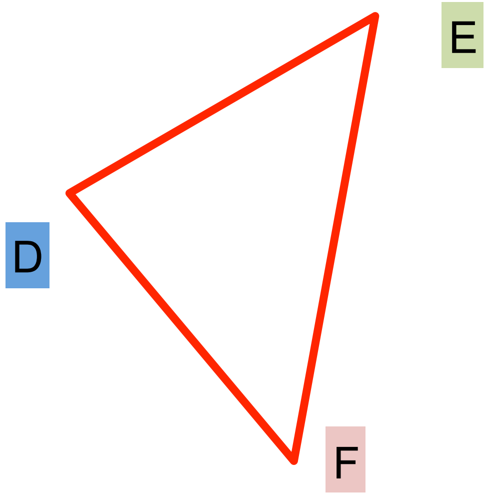


Determining unknown transformations

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 many degrees of freedom do we have? 6 = 3 (x,y) coordinates





Affine transform:
uniform scaling + shearing
+ rotation + translation

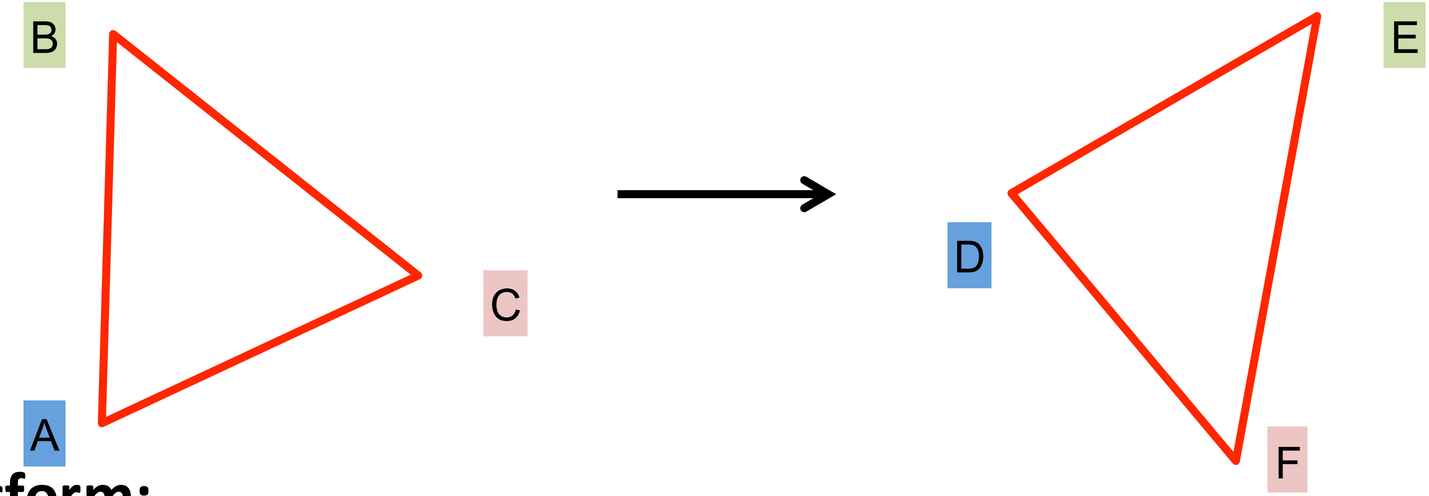
$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Imagine triangle as half a parallelogram

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 many degrees of freedom do we have?



Affine transform:

uniform scaling + shearing
+ rotation + translation

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

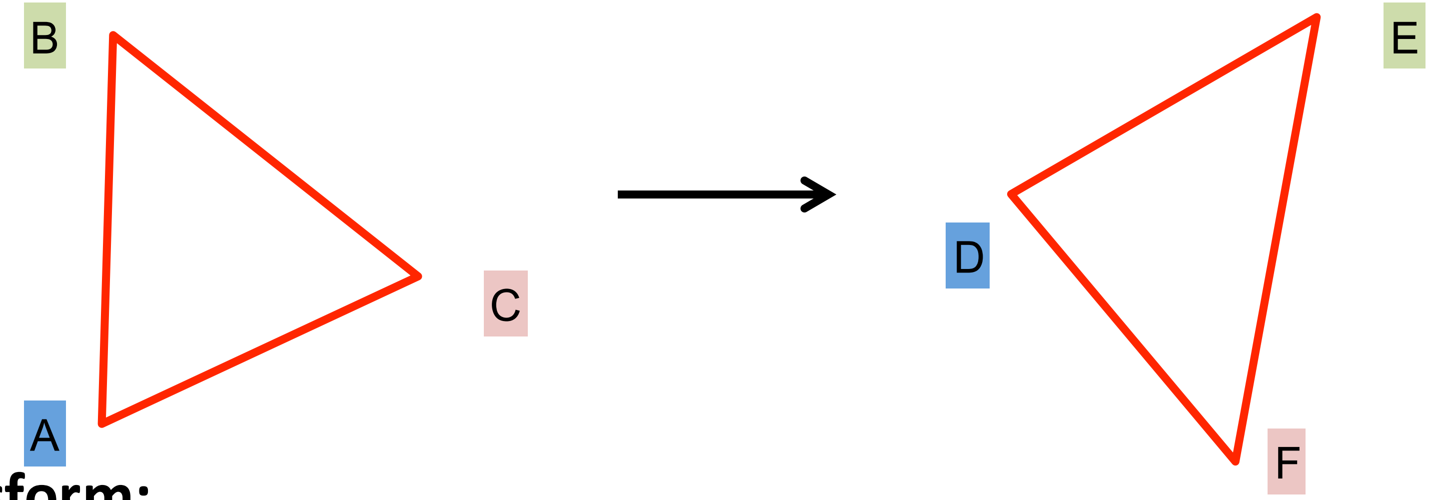
unknowns

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

point correspondences

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

Determining unknown transformations



Affine transform:
uniform scaling + shearing
+ rotation + translation

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

unknowns

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

point correspondences

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

1) Find pairs of corresponding points

$$(x_1, x_1')$$
$$(x_2, x_2')$$
$$(x_3, x_3')$$
$$\ldots$$

2) Write down an objective:

$$\min_M \sum_i \left\| x' - Mx \right\|^2$$

3) Solve for M

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:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \boxed{\quad ? \quad} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}$$

Stack equations from point correspondences:

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

Notation in system form:

$$\boldsymbol{b} = \mathbf{A} \boldsymbol{x}$$

Solving the linear system

Convert the system to a linear least-squares problem:

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

$$= (Ax - b)^T (Ax - b)$$

$$||x||^2 = x^T x$$

Example: $[x_1\ x_2\ ]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathrm{x}_1^2 + \mathrm{x}_2^2 = ||\mathrm{x}||^2$

Expand the error:

$$(AB)^T = B^T A^T$$

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

Minimize the error:

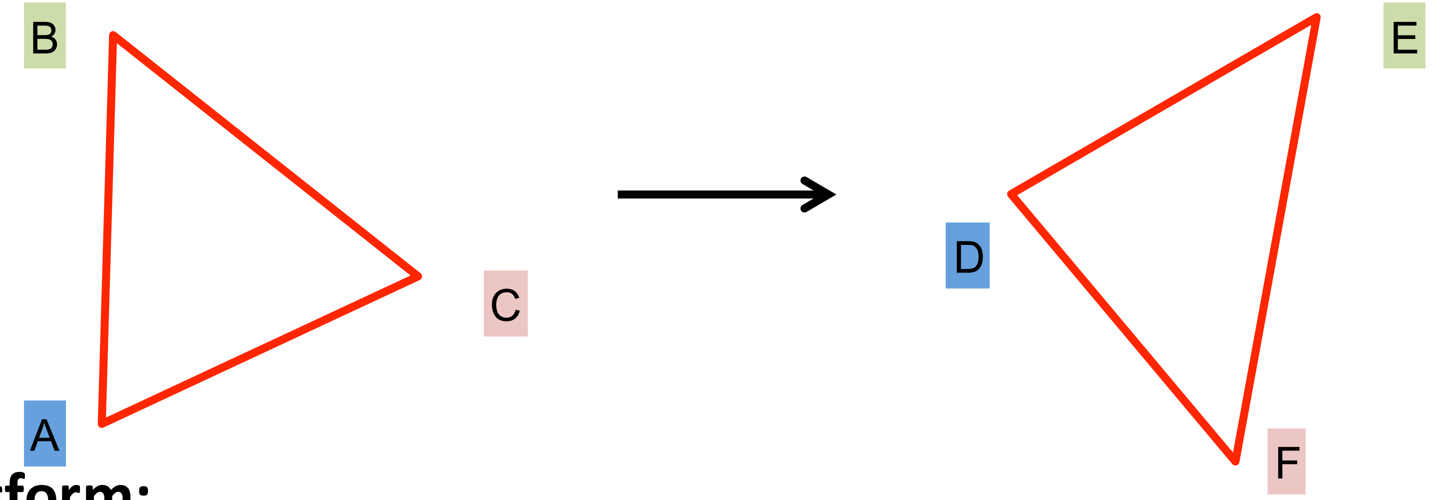
Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b}$

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

In Python:

```
x = numpy.linalg.
    solve(A, b)
```

Determining unknown transformations



Affine transform:
uniform scaling + shearing
+ rotation + translation

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

unknowns

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

point correspondences

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

1) Find pairs of corresponding points

$$(x_1, x_1')$$
$$(x_2, x_2')$$
$$(x_3, x_3')$$
$$\ldots$$

2) Write down an objective:

$$\min_M \sum_i ||x' - Mx||^2$$

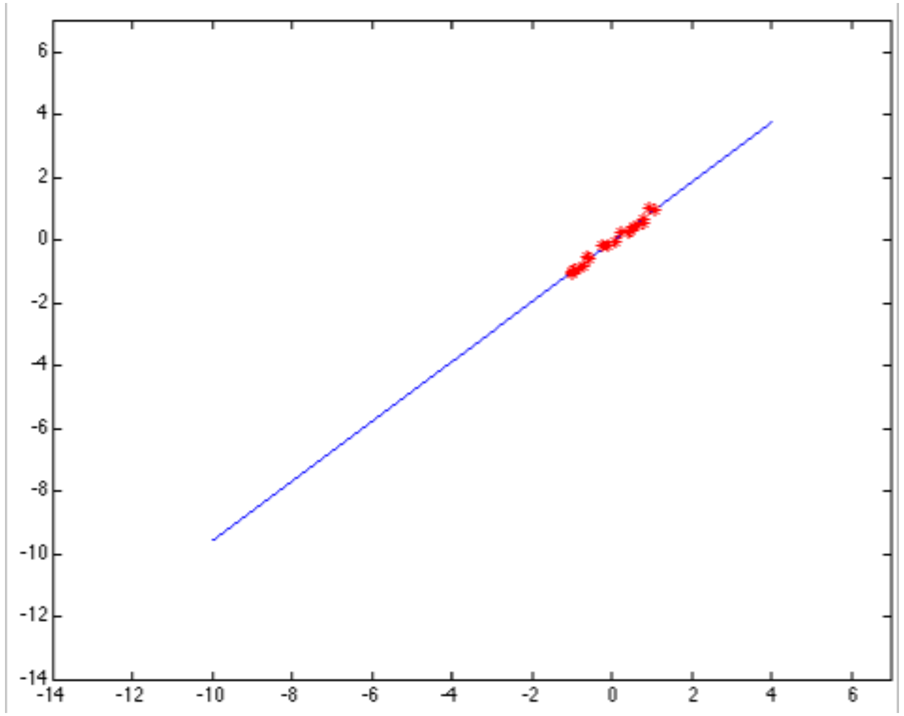
$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

3) Least squares:

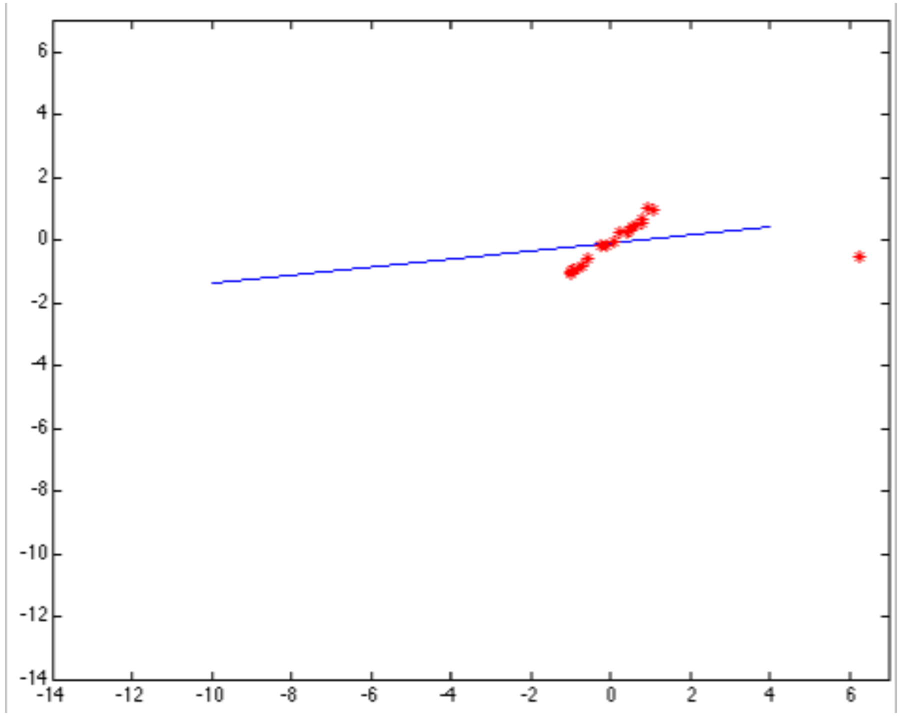
$$\boldsymbol{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \boldsymbol{b}$$

What are the issues with this approach?

Problems with noise



Least-squares error fit



Squared error heavily penalizes outliers

How did we fix this last time?

We will see next class another way to fix this for the task of finding a transform to match two images!