CSE 185 Introduction to Computer Vision

Lecture 5: Image Warping

Slides credit: Yuri Boykov, Ming-Hsuan Yang, Robert Collins,
Richard Szeliski, Steve Seitz, Alyosha Efros, Fei-Fei Li, etc.

Image Warping

http://www.jeffrey-martin.com

Image Warping (a.k.a. Domain Transforms)

- Parametric transformations
  - - Linear transformations of images via 2x2 matrices
  (a crash course on basic linear algebra)
  - Affine transformations
  - Homographies (3x3 transformation matrices)
- Estimation of parametric transformations (from corresponding points)
- Forward and inverse warps
  - - bilinear interpolation

Image Warping

- point processing: change ***range*** of image
  - $g(x) = T(f(x))$

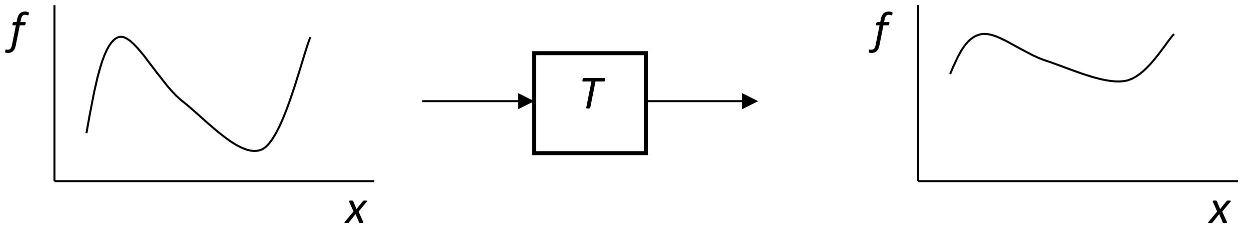


image warping: change ***domain*** of image

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

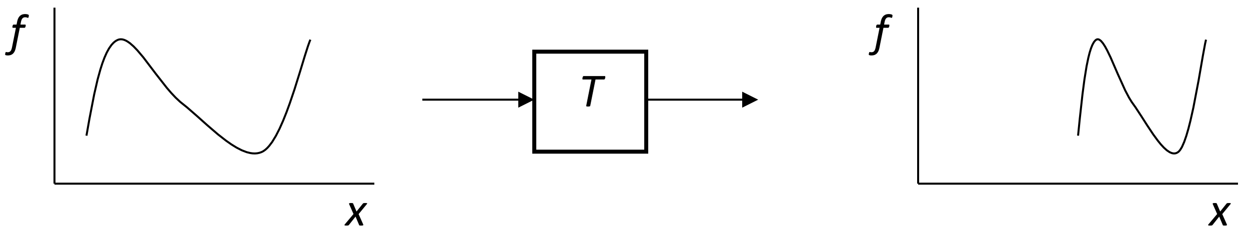


Image Warping

- point processing: change ***range*** of image
  - $g(x) = T(f(x))$

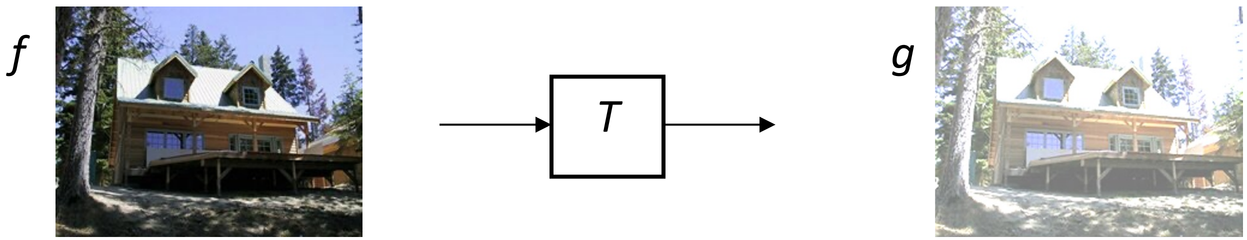
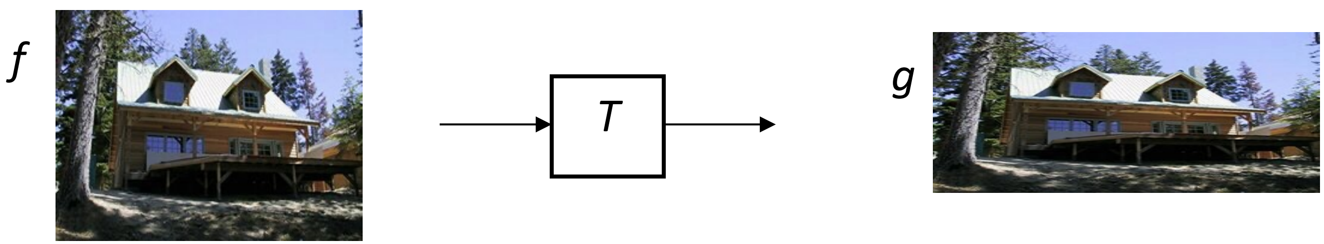


image warping: change ***domain*** of image

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



Parametric (global) warping

- Examples of parametric warps:

translation

rotation

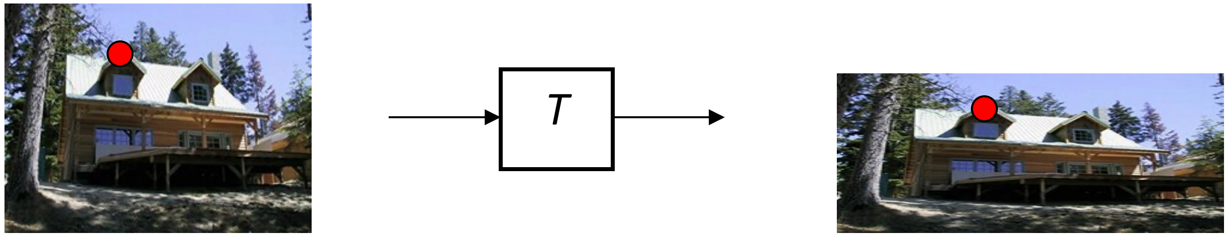
aspect

affine

perspective

cylindrical

Parametric (global) warping



p = (x,y) **p’** = (x’,y’)

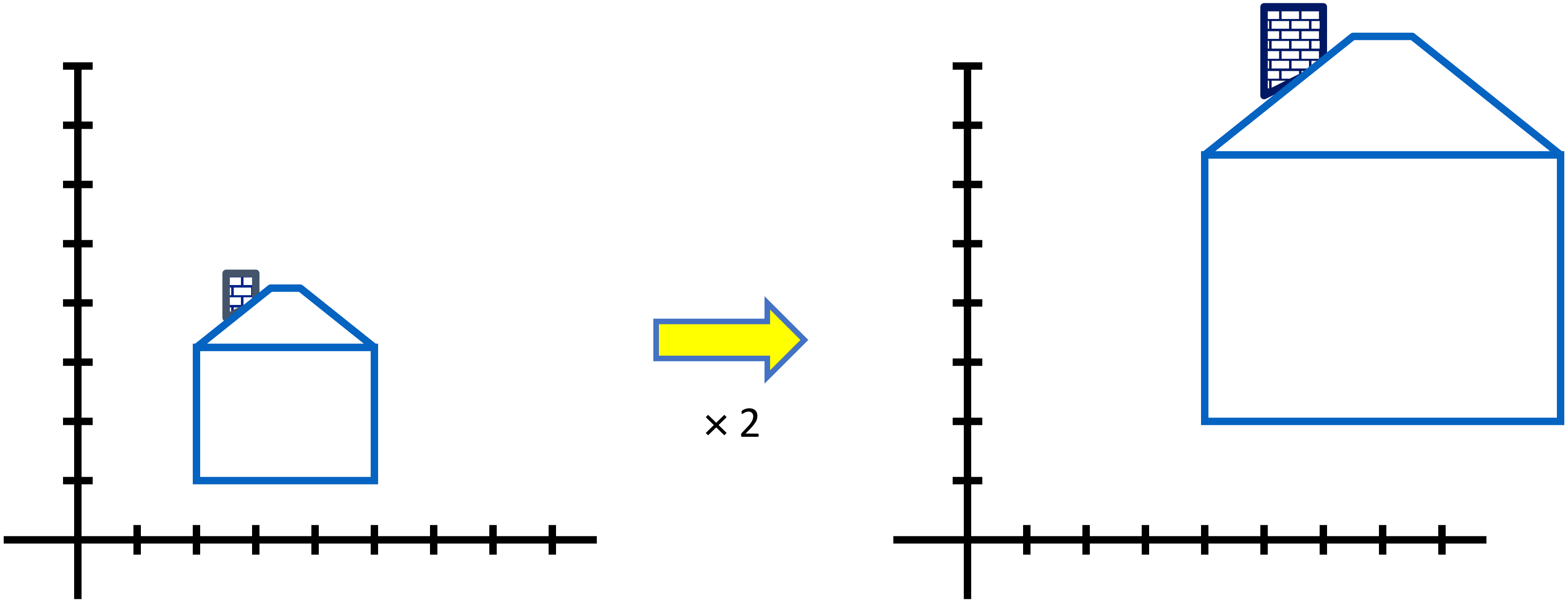
- Transformation T is a coordinate-changing machine:
- What does it mean that *T* is global?
  - the same transform for any point p
  - described by just a few numbers (parameters)
- Let’s represent *T* as a matrix: p’ = **M**p (linear transforms)

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

$$g(p') = g(M \cdot p) = f(p) = f(M^{-1} \cdot p')$$

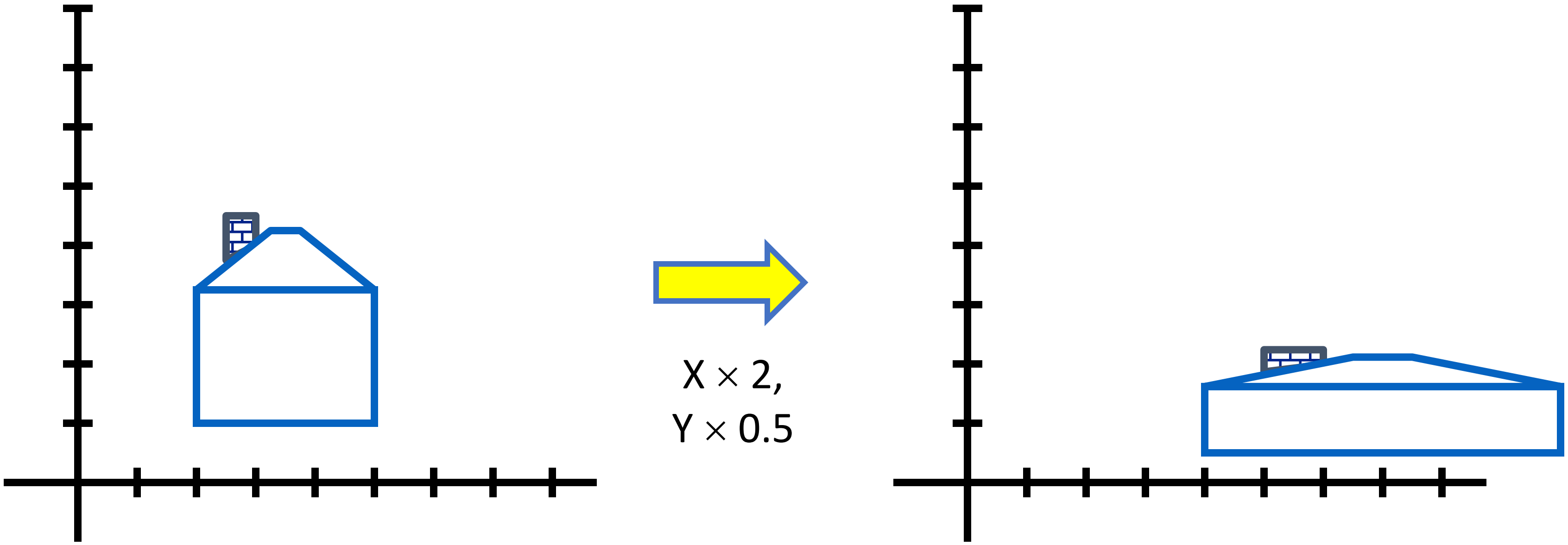
Scaling

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



Scaling

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



Scaling

- Scaling operation:

$$x' = ax$$

$$y' = by$$

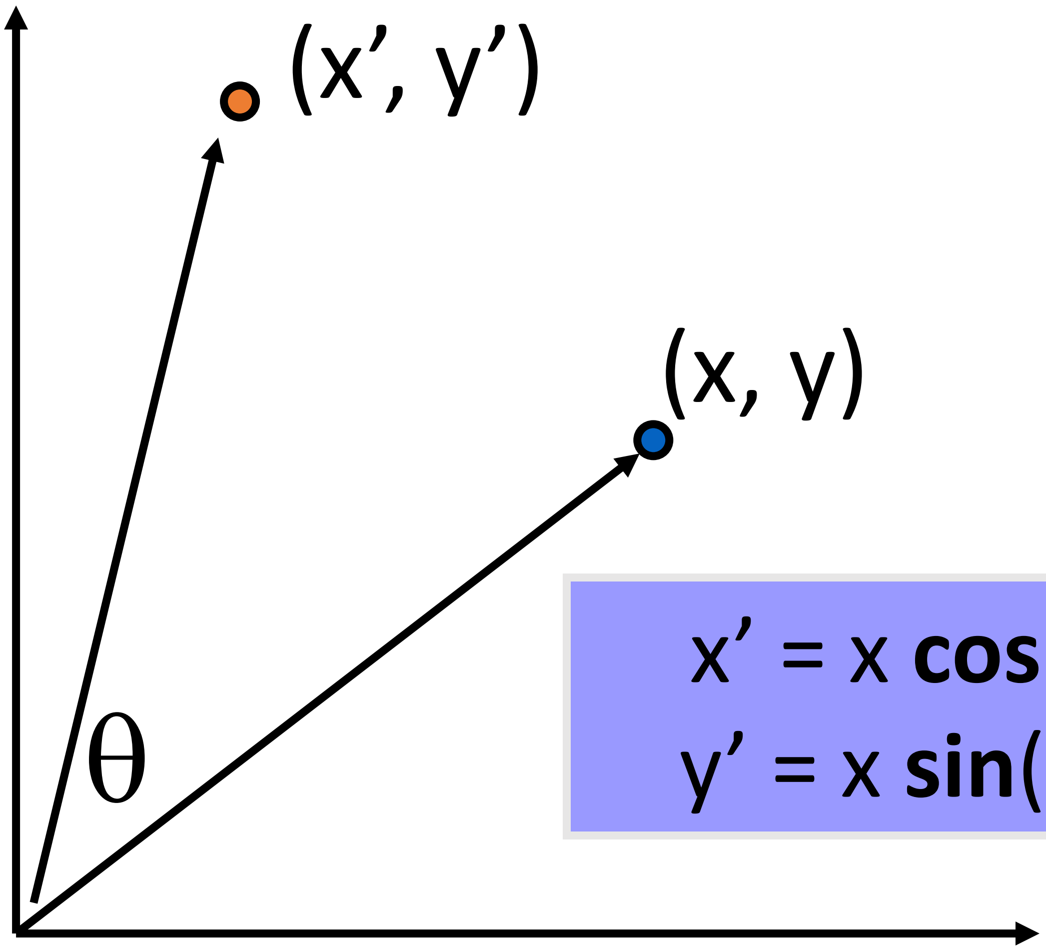
- Or, in matrix form:

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

scaling matrix S

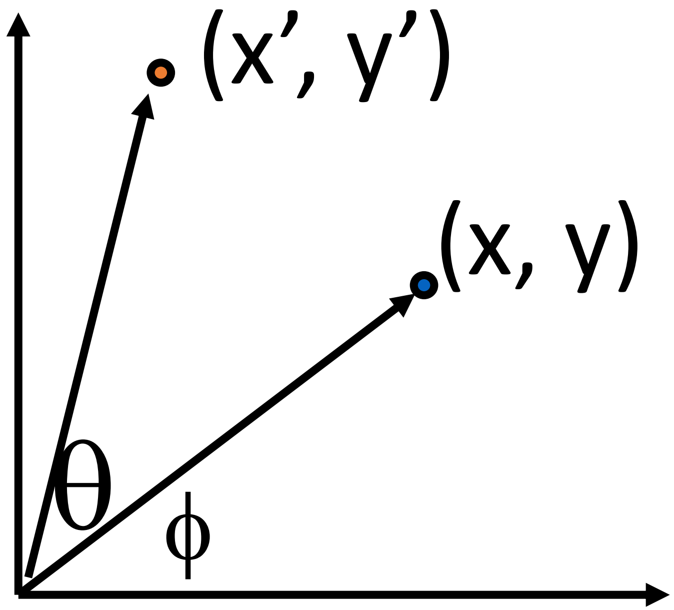
What's inverse of S?

2-D Rotation



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

2-D Rotation (derivation)



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

Trig Identity…

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

Substitute…

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

2-D Rotation

- This is easy to capture in matrix form:

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

- Even though $sin(\theta)$ and $cos(\theta)$ are nonlinear functions of θ,
  - ***x' is a linear combination of x and y***
  - ***y' is a linear combination of x and y***

- What is the inverse transformation?
  - Rotation by $-\theta$
  - For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

2x2 Matrices

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

2D Identity?

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

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

2D Scale around (0,0)?

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

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

2x2 Matrices

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

2D Rotate around (0,0)?

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

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

2D Shear?

$$x' = x + sh_x * y$$
$$y' = y$$

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

2x2 Matrices

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

2D Mirror about Y axis?

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

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

2D Mirror over (0,0)?

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

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

2x2 Matrices

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

2D Translation?

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

NO!

Only linear 2D transformations can be represented with a 2x2 matrix

All 2D Linear Transformations

- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror

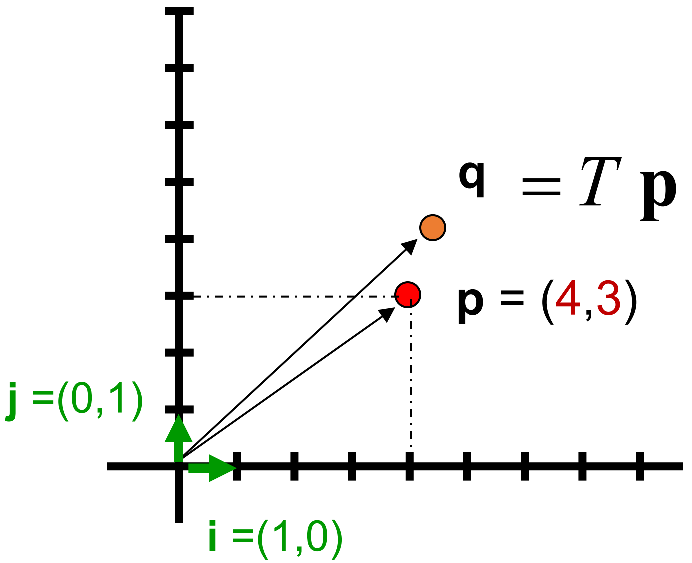
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Properties of linear transformations:
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Distance or length ratios are preserved on parallel lines
    - scaling of length/distances depends on (line) orientation only (see next slide)
  - Ratios of areas are preserved
  - Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} s & q \\ r & t \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

  - See pp. 40-41 of Hartley and Zisserman “Multiple View Geometry” (2nd edition)

Linear Transformation as **Space Deformation**



$$\underbrace{\begin{bmatrix} q_x \\ q_y \end{bmatrix}}_{\mathbf{q}} = \underbrace{\begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} 4 \\ 3 \end{bmatrix}}_{\mathbf{p}}$$

coordinates of ***q*** in basis **i,j**

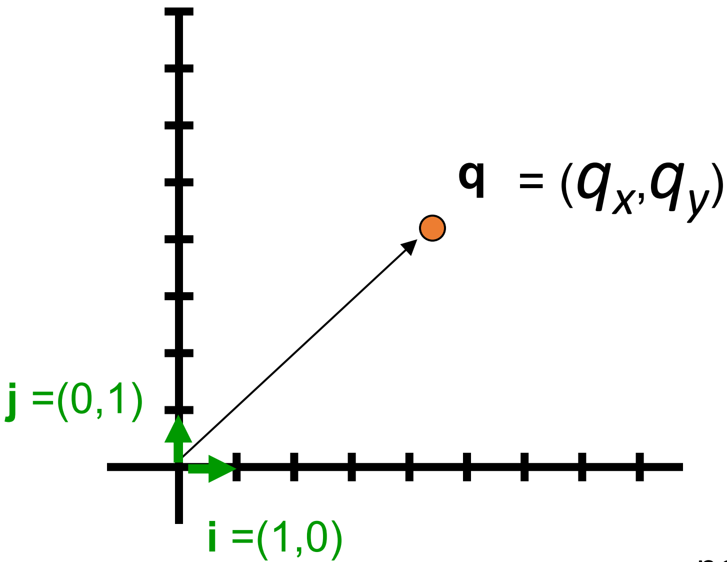
$$\mathbf{q} = q_x\,\mathbf{i} + q_y\,\mathbf{j}$$

coordinates of ***p*** in basis **i,j**

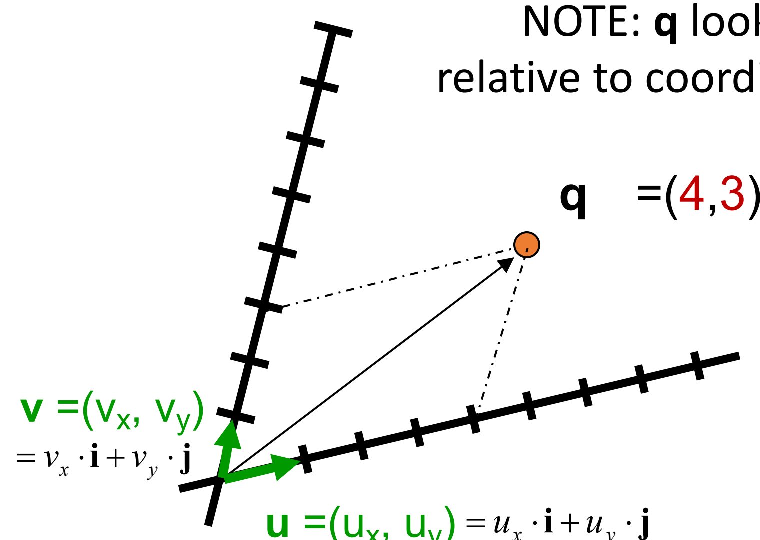
$$\mathbf{p} = 4\mathbf{i} + 3\mathbf{j}$$

point **p** is transformed into new point **q**

Linear Transformation as **Change of Basis**



NOTE: **q** looks just like **p** relative to coordinate system **u,v**



now interpret the columns of matrix T as some vectors **u** and **v** (their coordinates in basis **i**, **j**)

$$\begin{bmatrix} q_x \\ q_y \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

coordinates of ***q*** in basis **i,j**

$$\mathbf{q} = q_x\ \mathbf{i} + q_y\ \mathbf{j}$$

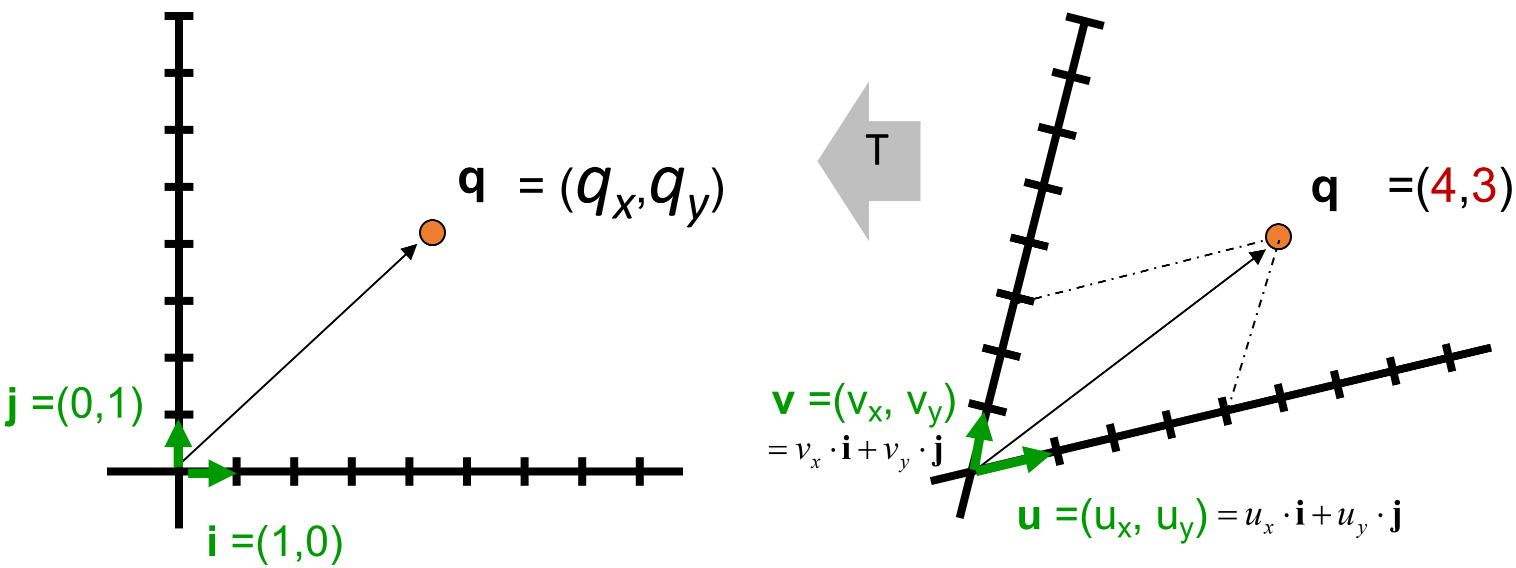
coordinates of ***q*** in basis **u,v**

$$\mathbf{q} = 4\mathbf{u} + 3\mathbf{v}$$

Indeed,

$$\mathbf{q} = 4\cdot(\overbrace{u_x\cdot\mathbf{i} + u_y\cdot\mathbf{j}}^{\mathbf{u}}) + 3\cdot(\overbrace{v_x\cdot\mathbf{i} + v_y\cdot\mathbf{j}}^{\mathbf{v}}) = (\overbrace{4\cdot u_x + 3\cdot v_x}^{q_x})\cdot\mathbf{i} + (\overbrace{4\cdot u_y + 3\cdot v_y}^{q_y})\cdot\mathbf{j}$$

Linear Transformation as **Change of Basis**



now interpret the columns of matrix T
as some vectors **u** and **v** (their coordinates in basis **i**, **j**)

$$\begin{bmatrix} q_x \\ q_y \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

coordinates of ***q*** in basis **i,j**

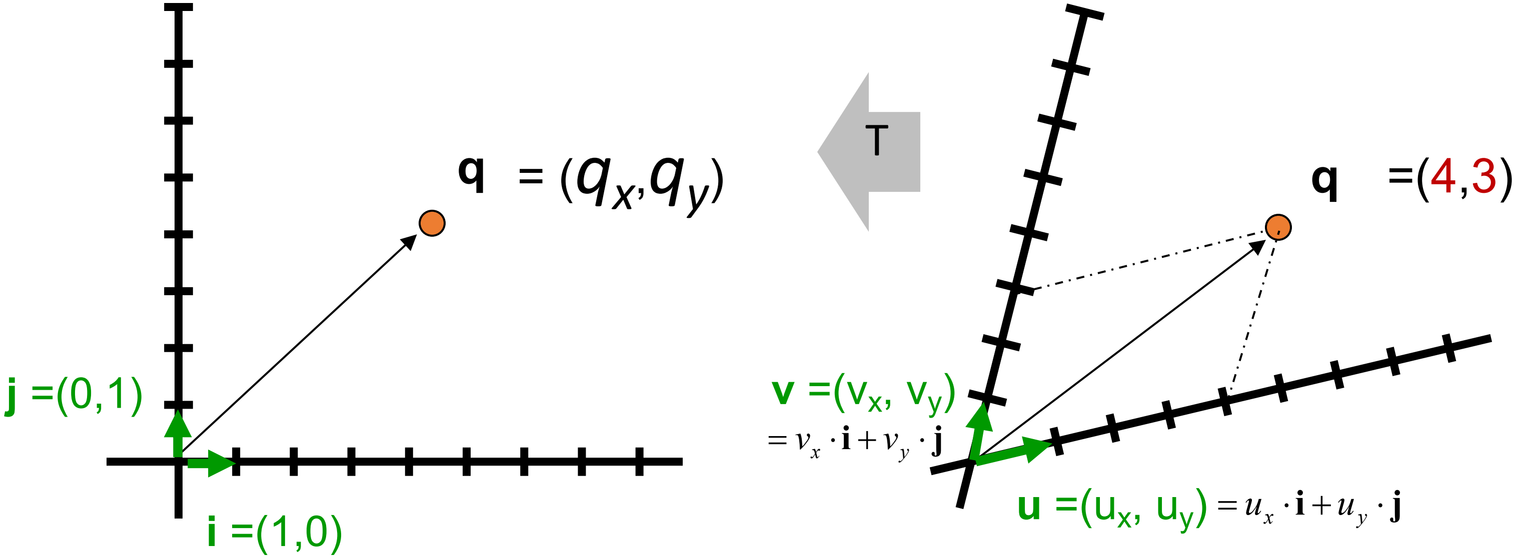
$$\mathbf{q} = q_x\,\mathbf{i} + q_y\,\mathbf{j}$$

coordinates of ***q*** in basis **u,v**

$$\mathbf{q} = 4\mathbf{u} + 3\mathbf{v}$$

point **q** represented in different coordinate systems

Linear Transformation as **Change of Basis**



now interpret the columns of matrix T
as some vectors **u** and **v** (their coordinates in basis **i, j**)

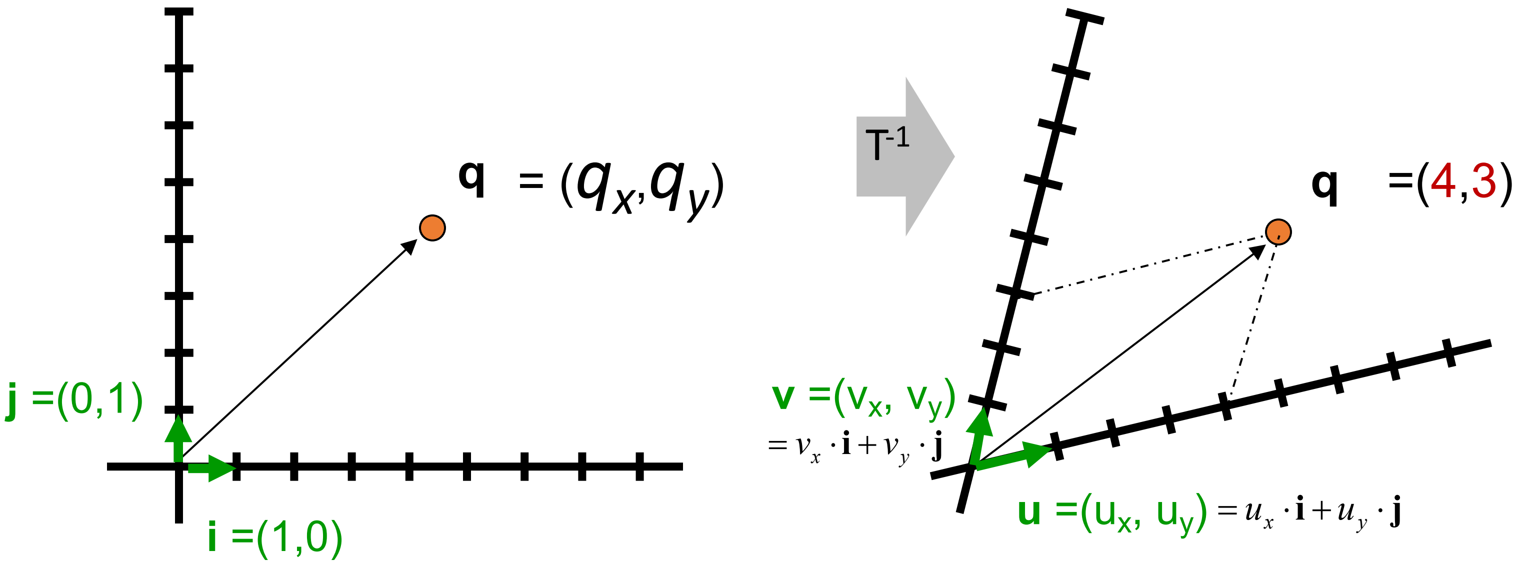
$$\begin{bmatrix} q_x \\ q_y \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

Any matrix can be seen as a (linear) coordinate system basis!!!

Question: What's the inverse matrix T^{-1} ?

$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \end{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix}$$

Linear Transformation as **Change of Basis**



now interpret the columns of matrix T
as some vectors **u** and **v** (their coordinates in basis **i, j**)

$$\begin{bmatrix} q_x \\ q_y \end{bmatrix} = \begin{bmatrix} u_x & v_x \\ u_y & v_y \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

Any matrix can be seen as a (linear) coordinate system basis!!!

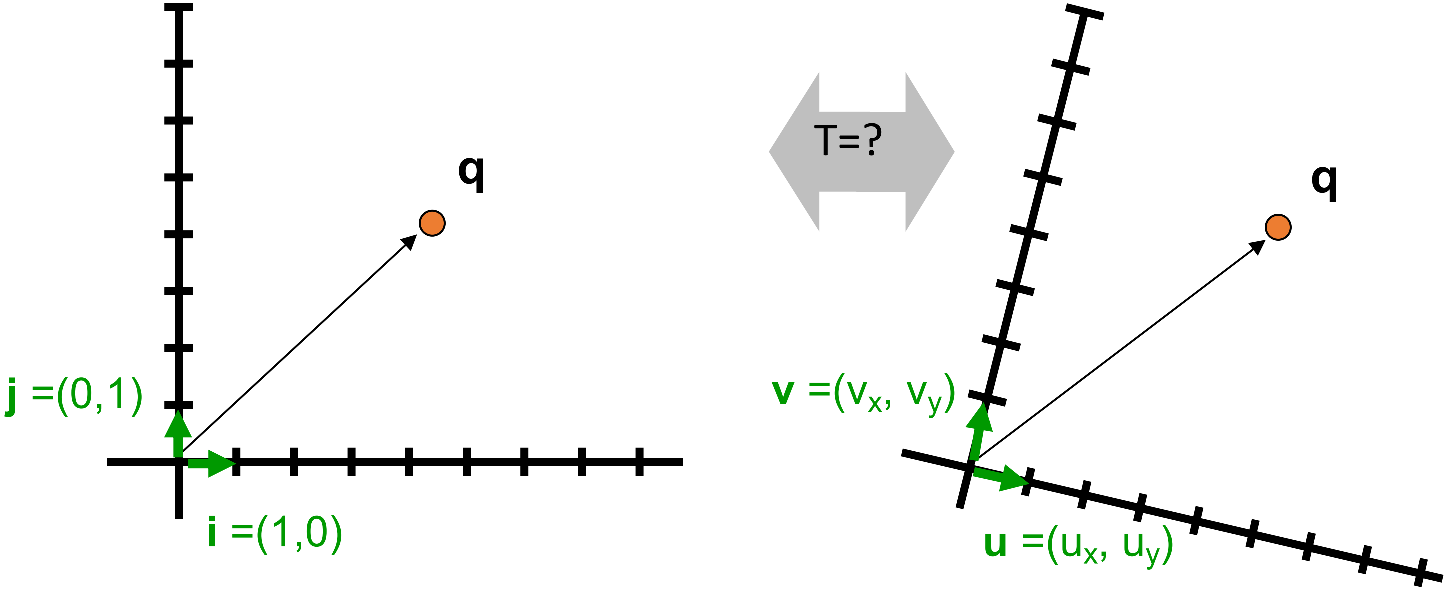
Question: What's the inverse matrix T^{-1} ?

$$\begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} i_x & j_x \\ i_y & j_y \end{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix}$$

coordinates of **i** and **j** in basis **u,v**

$$\mathbf{i} = i_x \cdot \mathbf{u} + i_y \cdot \mathbf{v}$$
$$\mathbf{j} = j_x \cdot \mathbf{u} + j_y \cdot \mathbf{v}$$

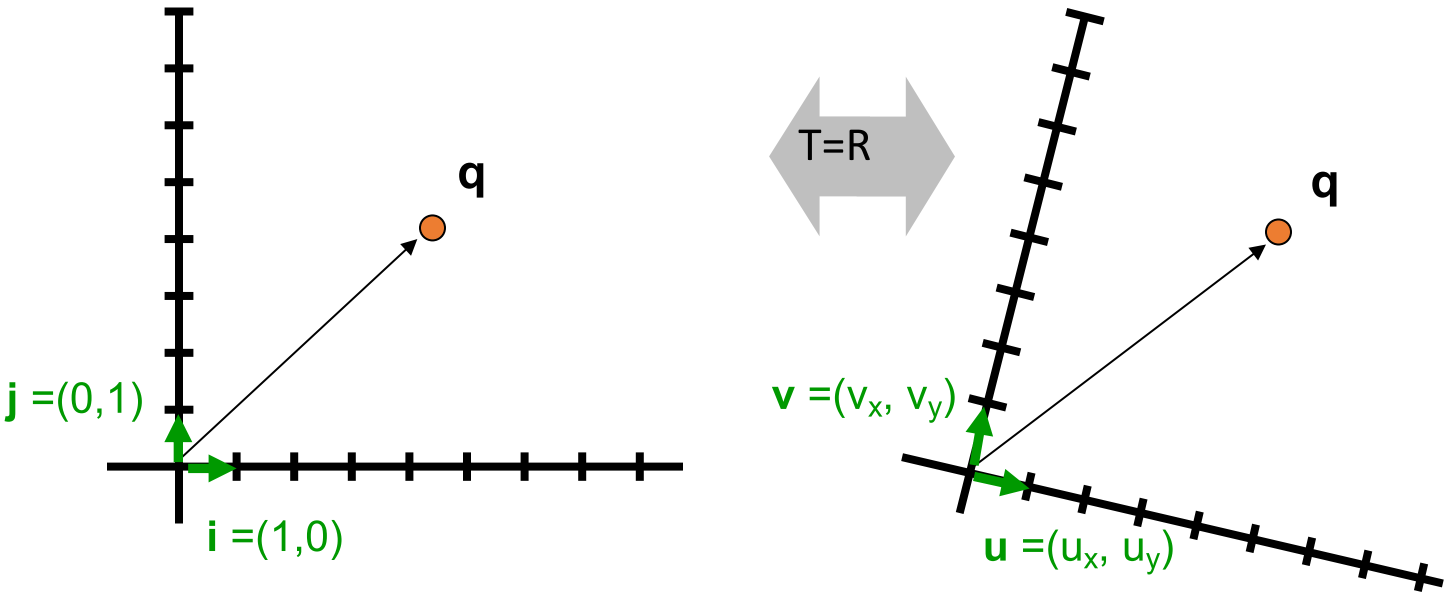
Linear Transformation as **Change of Basis**



Any matrix can be seen as a (linear) coordinate system basis!!!

Question: What is T if both coordinate systems have **ortho-normal basis**?

Linear Transformation as **Change of Basis**



Any matrix can be seen as a (linear) coordinate system basis!!!

Question: What is T if both coordinate systems have **ortho-normal basis**?

Then matrix T represents ***rotation***, ***reflection***, or their combination (***rotoreflection***) of the coordinate basis

Towards Homogeneous Coordinates

❑ **Q: Can we represent translation by matrix multiplication?**

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

very simple, but
not a *linear* transformation *in 2D*

$$T(p+q) \neq T(p) + T(q)$$
$$T(\lambda p) \neq \lambda T(p)$$

Answer: Yes, using homogeneous coordinates and 3x3 matrices

Homogeneous coordinates

- represent coordinates in 2 dimensions with a 3-vector

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

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

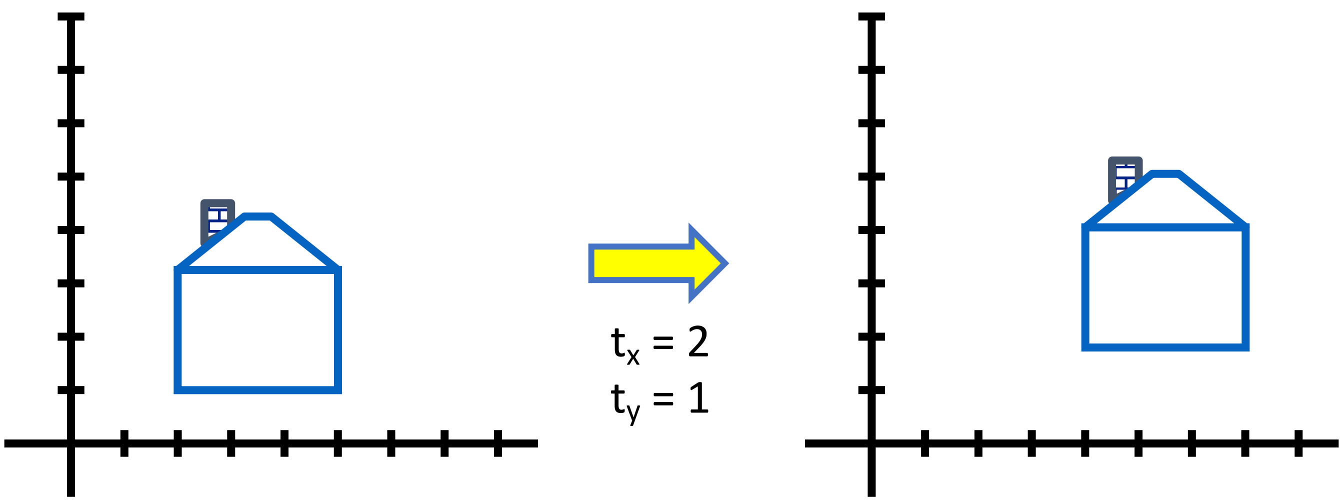
Translation matrix (3x3)

Translation

- Example of translation

Homogeneous Coordinates

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

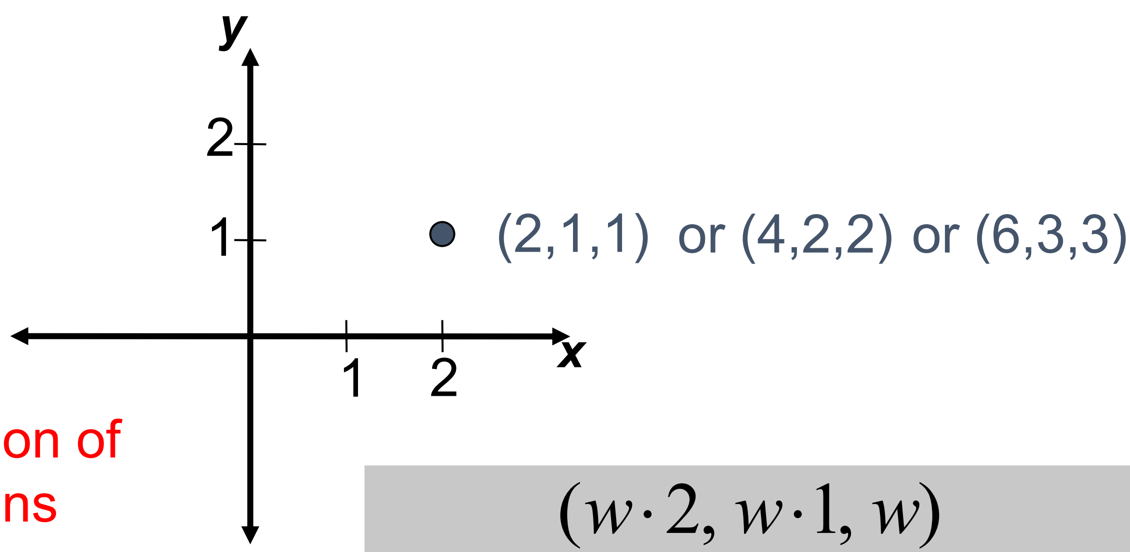


Homogeneous Coordinates (in general)

- Add a 3rd coordinate to every 2D point
  - (x, y, w) represents a point at location (x/w, y/w)
  - (0, 0, 0) is not allowed

Advantages of homogeneous coordinate system:

- simple matrix representation of many useful transformations

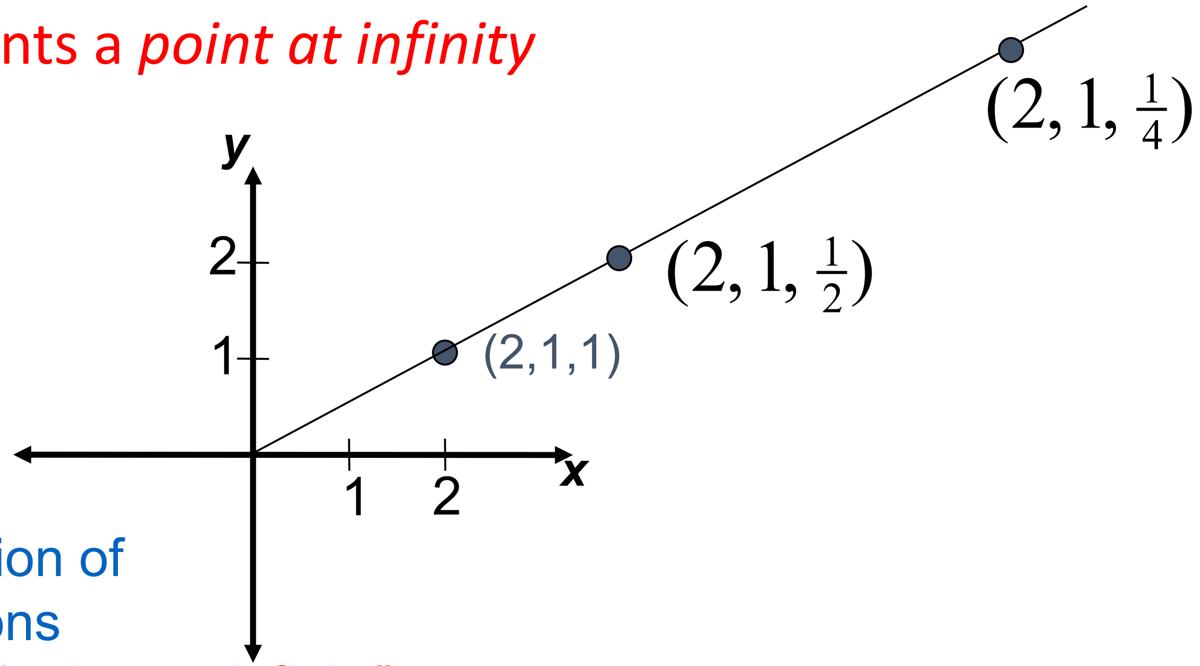


$$(w\cdot 2, w\cdot 1, w)$$

represent the same 2D point for any value of *w*

Homogeneous Coordinates (in general)

- Add a 3rd coordinate to every 2D point
  - (x, y, w) represents a point at location (x/w, y/w)
  - (0, 0, 0) is not allowed
  - (x, y, 0) represents a *point at infinity*



Advantages of homogeneous coordinate system:

- simple matrix representation of many useful transformations
- allows to expand R^2 with "*points at infinity*" (like $\pm\infty$ for R^1) using finite numerical representation

Basic 2D Transformations via 3x3 matrices

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

Translate

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

Scale

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

Rotate

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

Shear

all of the above are special cases of a general Affine Transformation:

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

Composing Affine Transformations

- **Example**:

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

composition of any affine transforms is still affine

(as easy to check)

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

p' = $T(t_x,t_y)$ $R(\Theta)$ $S(s_x,s_y)$ **p**

- **In general**: any affine transformation is a combination of translation, rotation/reflection, and anisotropic scaling

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 …
  - Linear 2D transformations, and
  - Translations
- Properties of affine transformations:
  - Origin does not necessarily map to origin (**new** compared to 2x2 matrices)
  - Lines map to lines
  - Parallel lines remain parallel
  - Length/distance ratios are preserved on parallel lines
  - Ratios of areas are preserved
  - Closed under composition

Projective Transformations (a.k.a. *homographies*)

transformations in homogeneous coordinate space via general 3x3 matrices

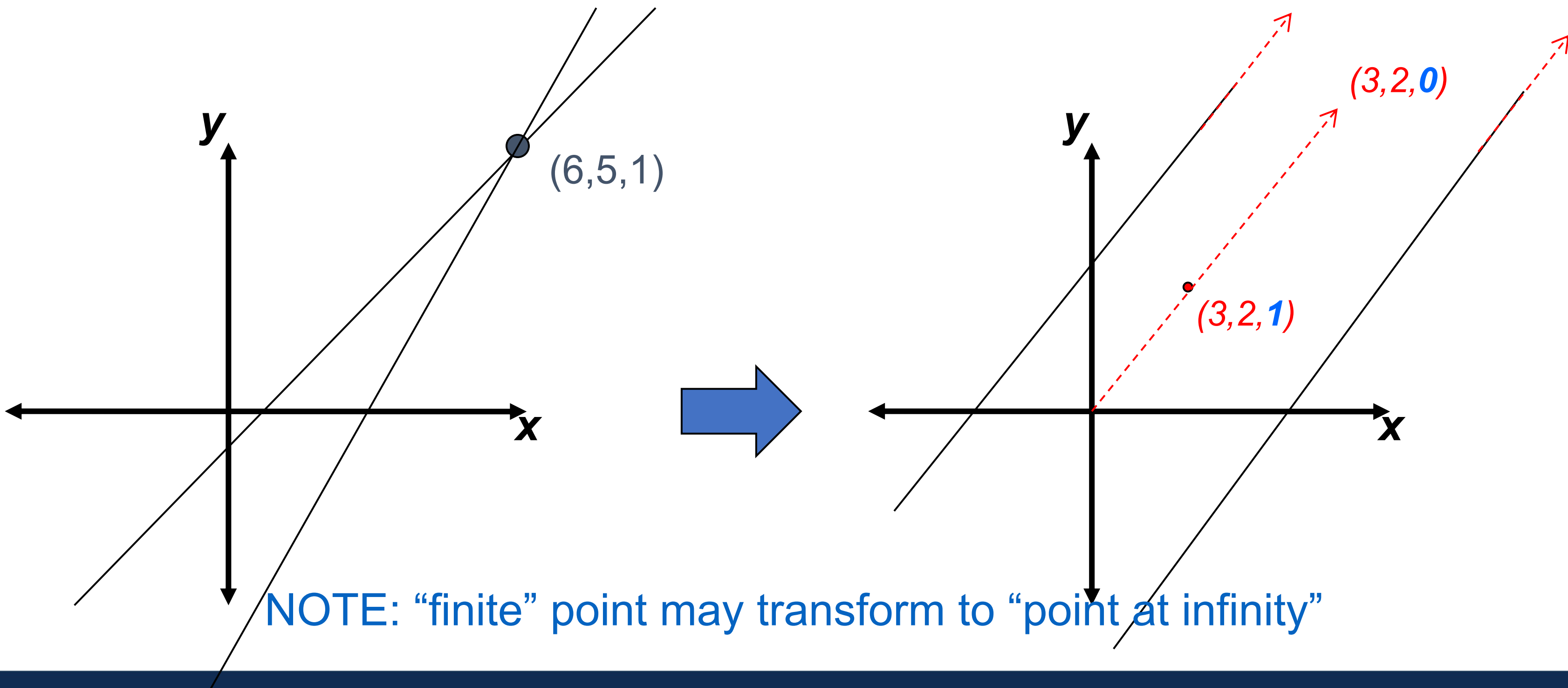
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Projective transformations …
  - Affine transformations, and
  - Projective warps
- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines (indeed, line of hom. points **p** means **a**·**p**=0 for some **a**. Then, **b**·H**p**=0 for **b**=**a**H^{-1})
  - Parallel lines do not necessarily remain parallel
  - Non-parallel lines may become parallel
  - Distance/length or area ratios are not preserved
  - Closed under composition

Projective Transformations (a.k.a. *homographies*)

- ❑ Parallel lines do not necessarily remain parallel
- ❑ <u>Non-parallel lines may become parallel</u>

$$\begin{bmatrix}3\\2\\0\end{bmatrix} = \underbrace{\begin{bmatrix}a & b & c\\ d & e & f\\ -1 & 1 & 1\end{bmatrix}}_{H}\begin{bmatrix}6\\5\\1\end{bmatrix}$$

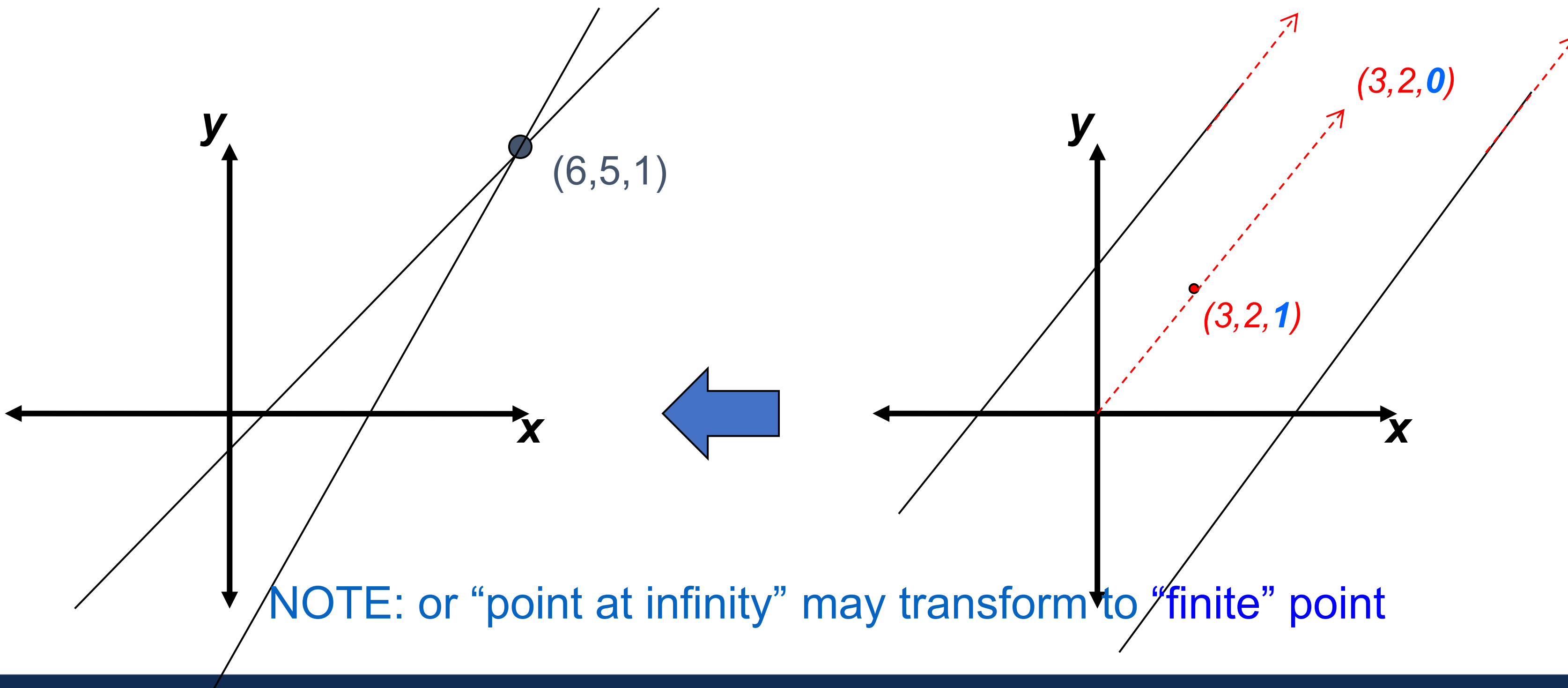


NOTE: "finite" point may transform to "point at infinity"

Projective Transformations (a.k.a. *homographies*)

- Parallel lines do not necessarily remain parallel
- Non-parallel lines may become parallel

$$\begin{bmatrix} 12 \\ 10 \\ 2 \end{bmatrix} = \underbrace{\begin{bmatrix} a' & b' & c' \\ d' & e' & f' \\ g' & h' & i' \end{bmatrix}}_{H^{-1}} \begin{bmatrix} 3 \\ 2 \\ 0 \end{bmatrix}$$

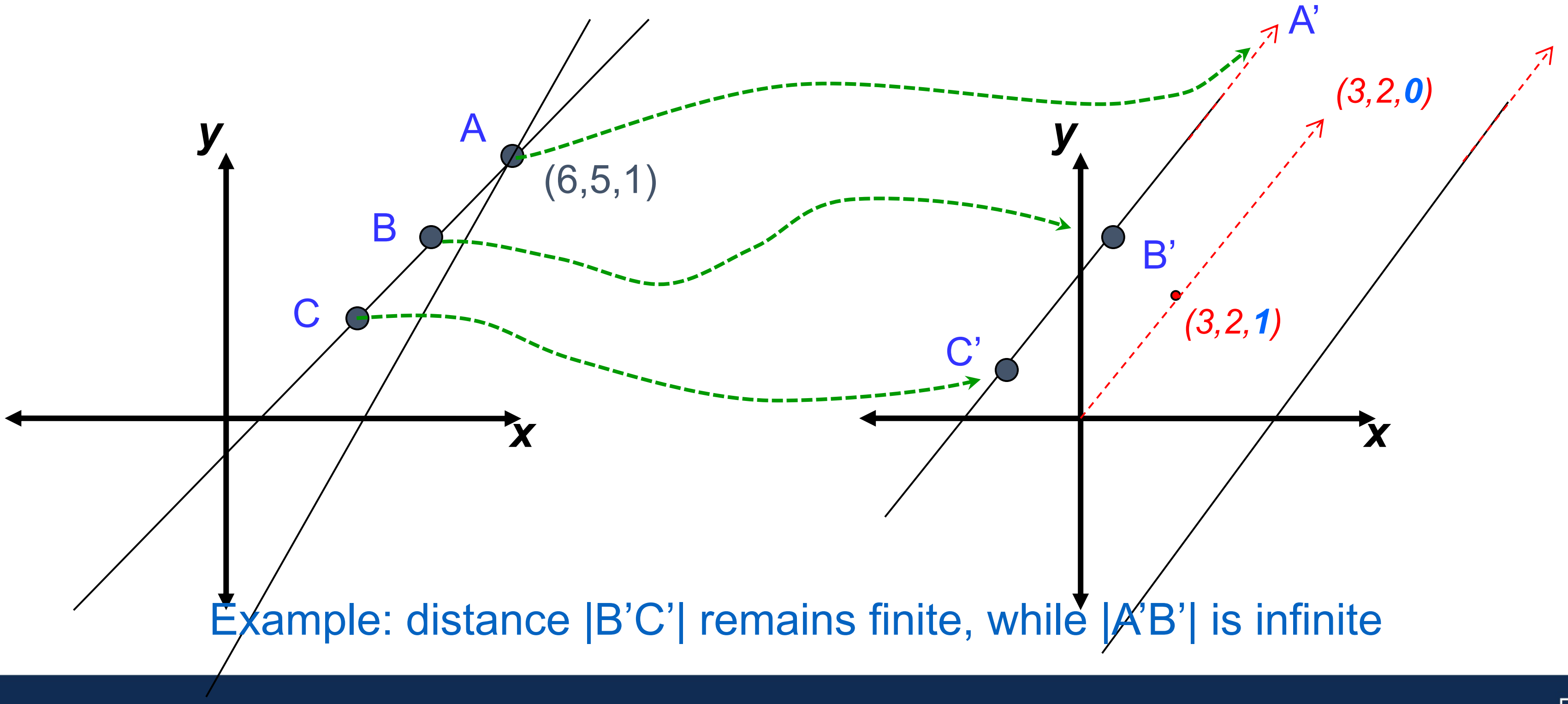


NOTE: or “point at infinity” may transform to “finite” point

Projective Transformations (a.k.a. *homographies*)

❑ Distance/length or area ratios are not preserved

$$\begin{bmatrix} 3 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 6 \\ 5 \\ 1 \end{bmatrix}$$



Example: distance |B'C'| remains finite, while |A'B'| is infinite

Projective Transformations (a.k.a. *homographies*)

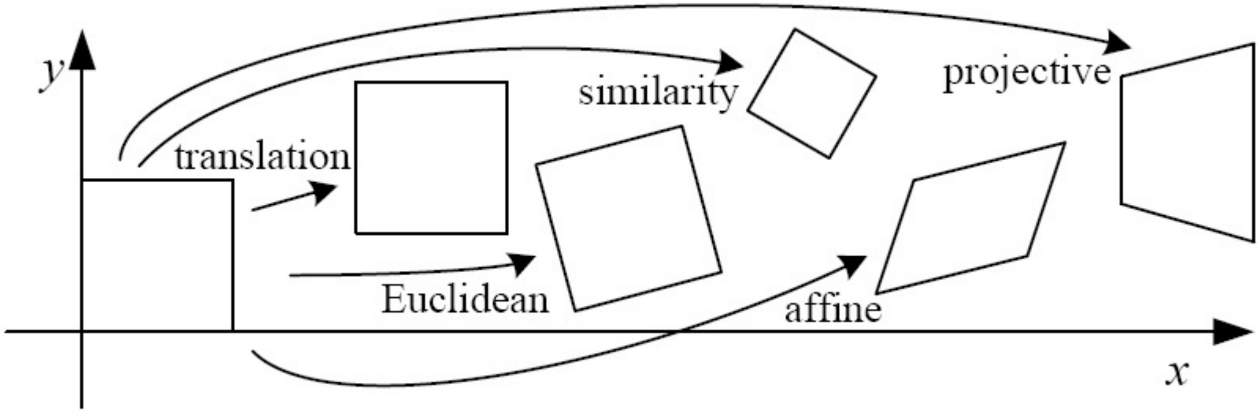
- General property to keep in mind (Theorem 2.10 from Hartley&Zisserman)

- An invertible mapping h from a (homogeneous) plane P^2 onto P^2 preserves straight lines **iff** there exists a non-singular 3x3 matrix H s.t.

- $$h(x) = H \cdot x$$ for any $x \in P^2$

That is, any transformation of a plane onto a plane that preserves straight lines must be a *homography*.

2D image transformations



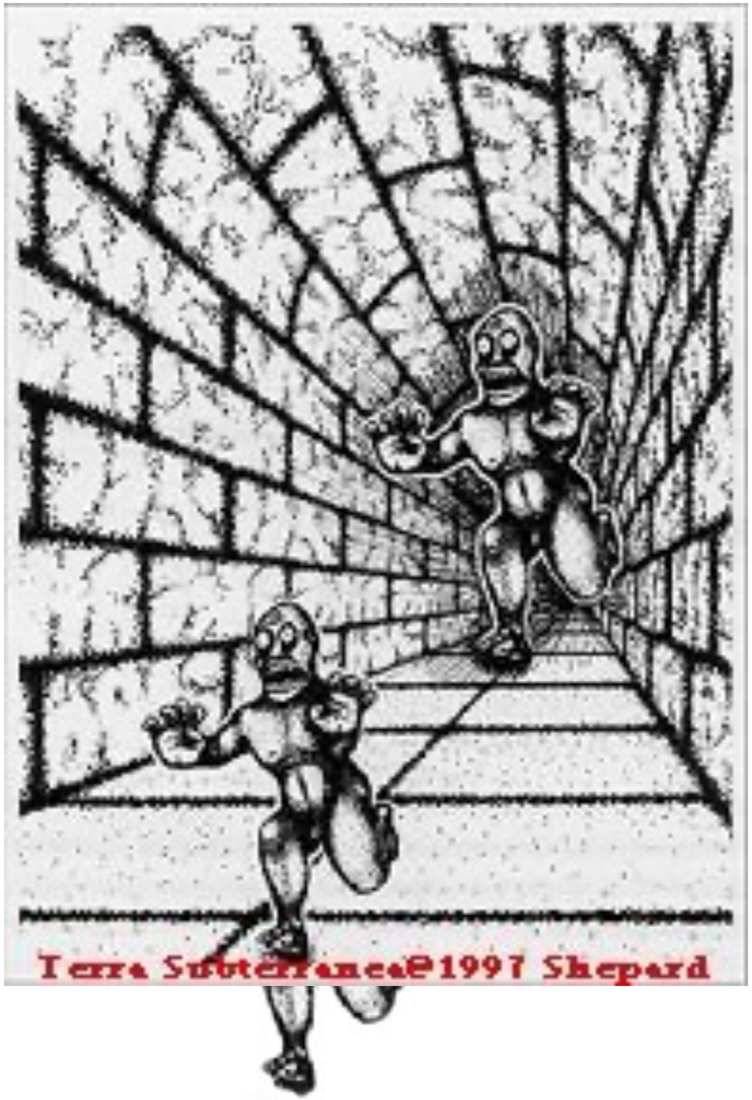
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R}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

See Hartley and Zisserman, p. 44

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Q: What best describes the transformation between two monsters in this image?



A: translation

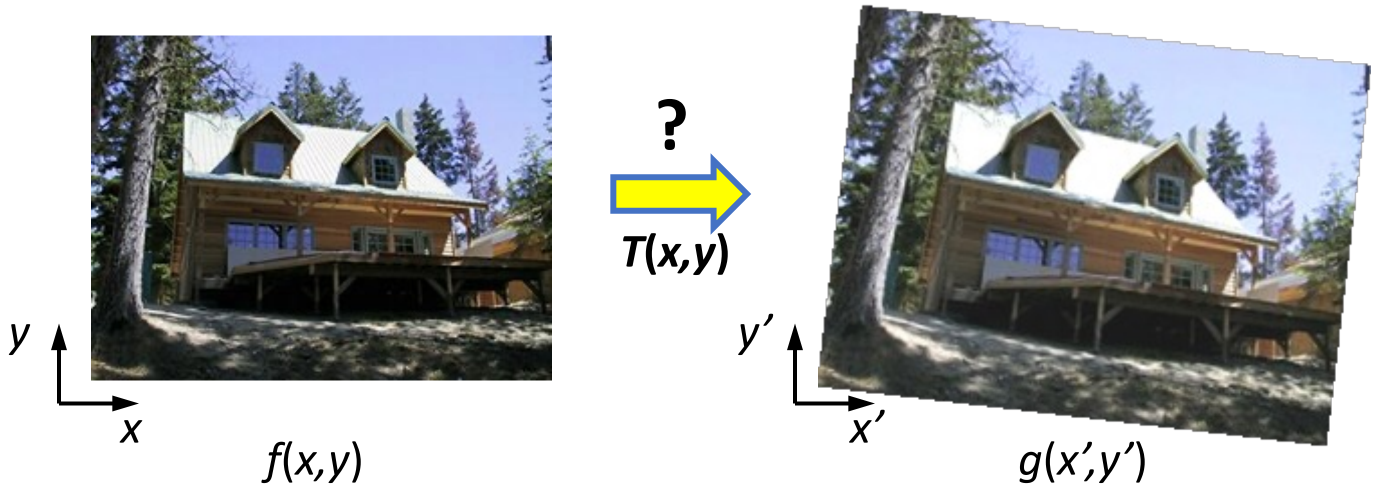
B: translation + scale

C: projective

Remaining parts of this lecture

- Estimation of parametric transformations (from corresponding points)
- Forward and inverse warps

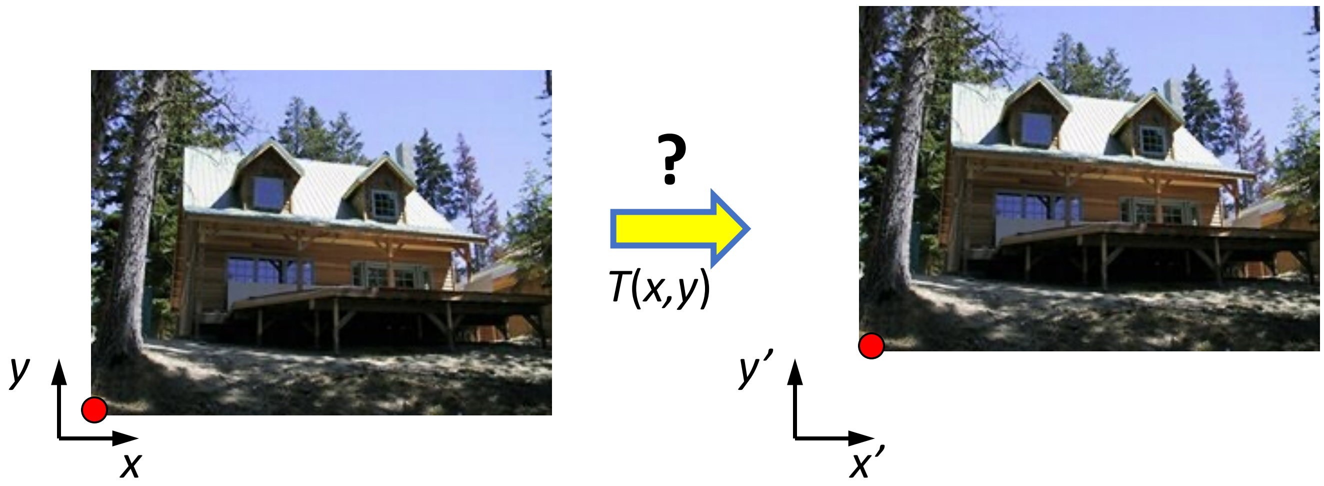
Recovering Parametric Transformations



- What if we know *f* and *g* and want to recover transform T?
  - e.g. to better align images (**image registration**)
  - willing to let user provide correspondences

Q: How many pairs of corresponding points do we need?

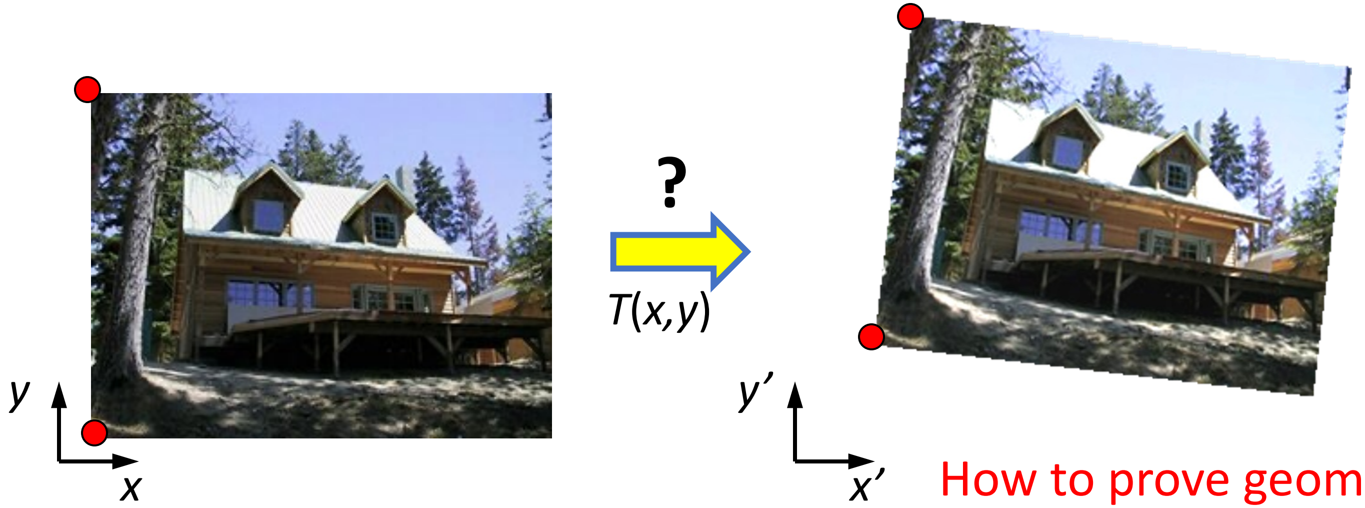
Translation: # correspondences?



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

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

Euclidian: # correspondences?



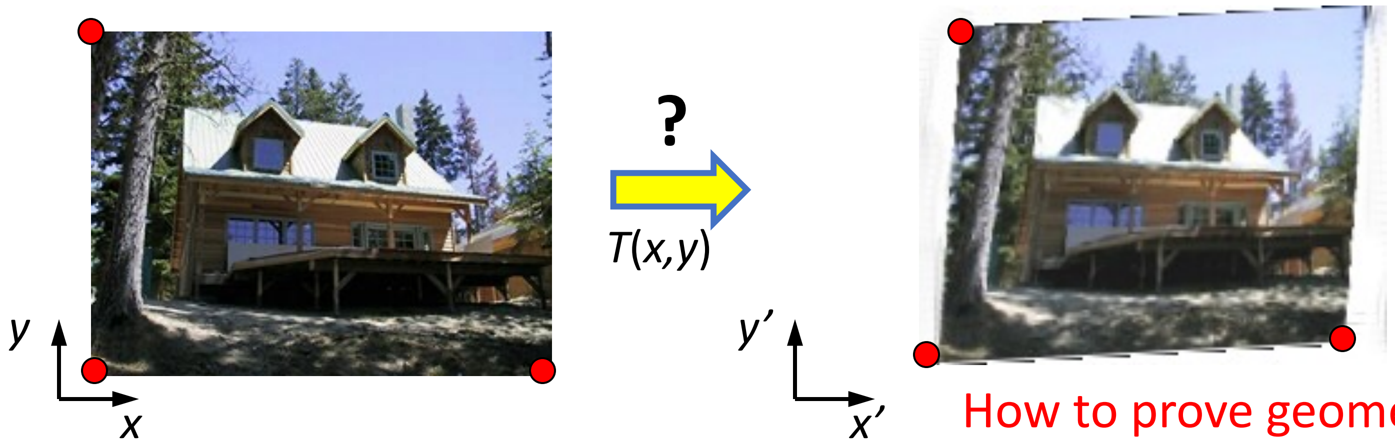
How to prove geometrically that 2 pairs is enough?

(use rigid transformation invariants to map an arbitrary point)

- ❑How many correspondences needed for translation+rotation?
- ❑How many DOF?
- ❑Transformation matrix?

$$\mathbf{M} = \begin{bmatrix} \cos\theta & -\sin\theta & c_x \\ \sin\theta & \cos\theta & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Affine: # correspondences?



How to prove geometrically that 3 pairs is enough?

(use affine transformation invariants to map an arbitrary point)

- ❑How many correspondences needed for affine?
- ❑How many DOF?
- ❑Transformation matrix?

$$\mathbf{M} = \begin{bmatrix} a & b & c \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

Algebraic point of view

$$\mathbf{p}'_i = M\ \mathbf{p}_i \qquad \underbrace{\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix}}_{\mathbf{p}'_i} = \underbrace{\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}}_{\mathbf{p}_i}$$

for any given pair of corresponding points $(\mathbf{p}_i, \mathbf{p}'_i)$

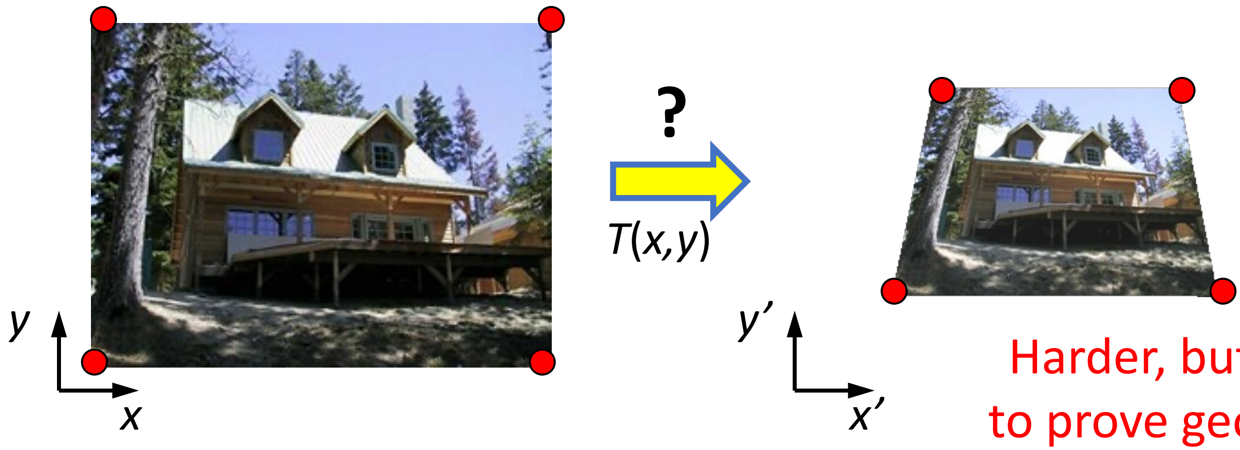
$$\Rightarrow \begin{cases} x'_i = a x_i + b y_i + c \\ y'_i = d x_i + e y_i + f \end{cases}$$

6 unknown parameters (variables)

Each pair of corresponding points $(\mathbf{p}_i, \mathbf{p}'_i)$ gives **two linear equations** w.r.t 6 unknown coefficients of matrix *M* with known point coordinates for $\mathbf{p}_i$ and $\mathbf{p}'_i$

3 pairs of corresponding points give 3x2 (=6) linear equations allowing to **resolve 6 unknown parameters**

Projective: # correspondences?

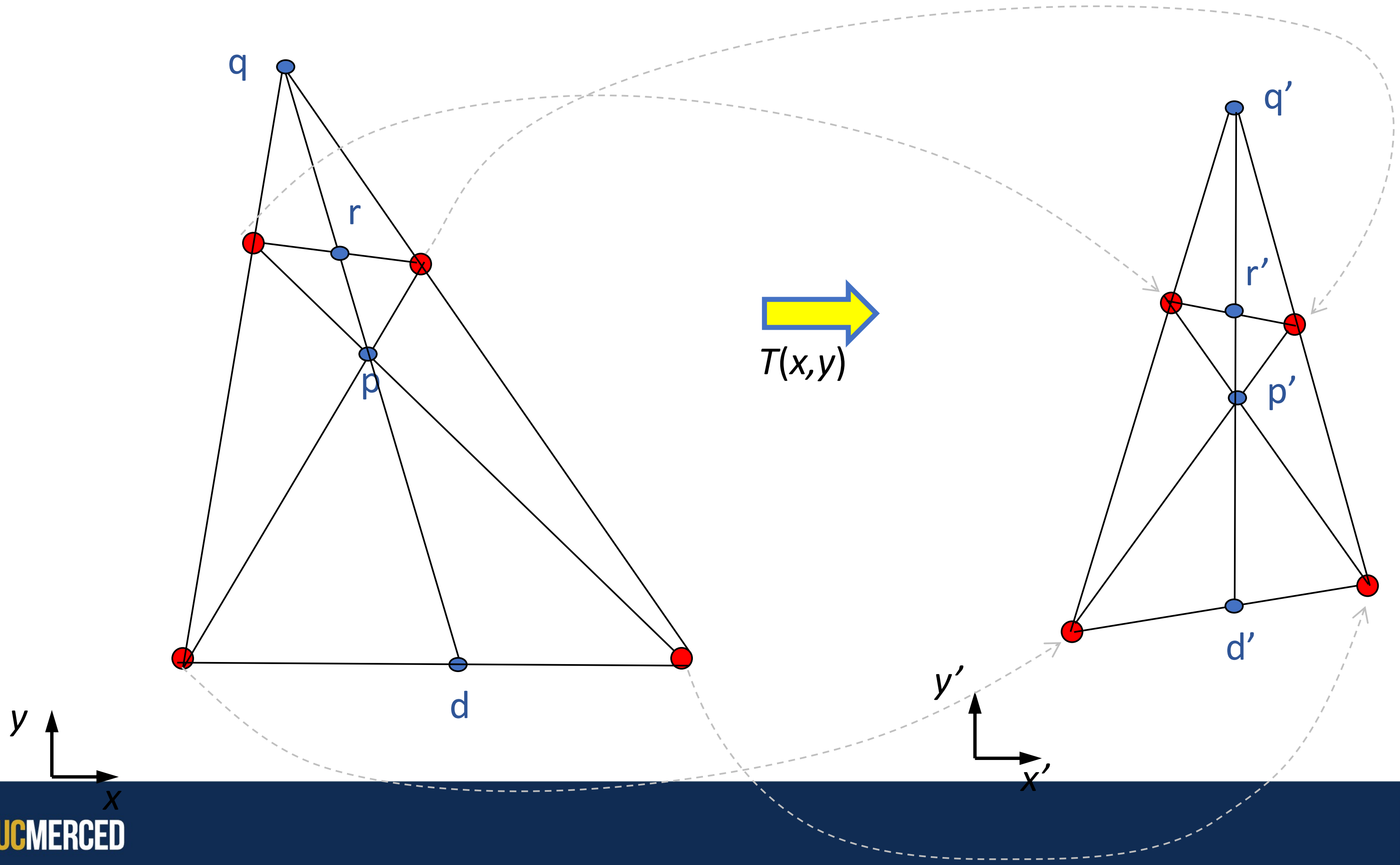


Harder, but possible
to prove geometrically
that 4 pairs is enough.
(can use only *line preservation*)

- ❑How many correspondences needed for projective?
- ❑How many DOF?
- ❑Transformation matrix?

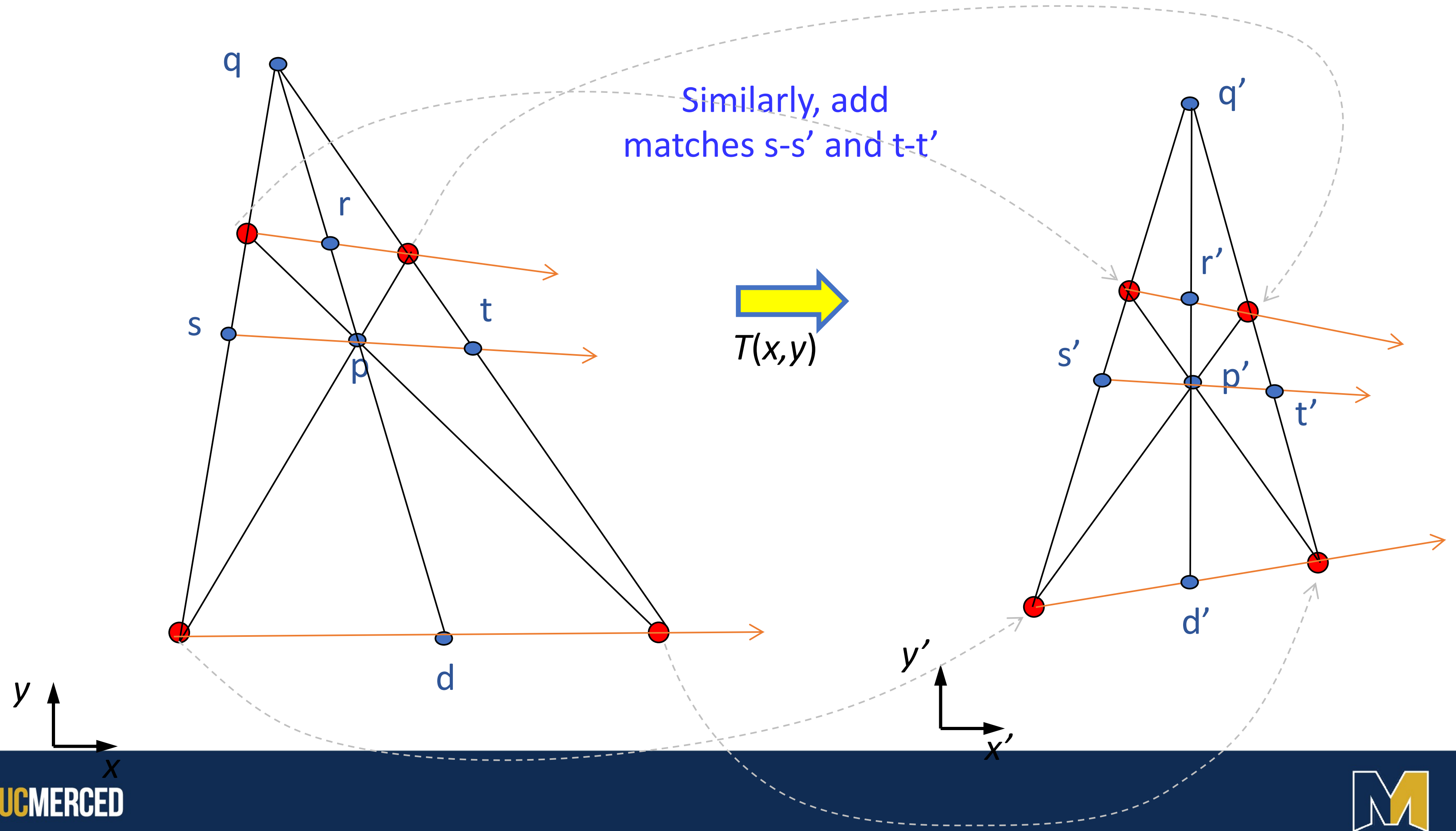
Projective: # correspondences?

4 matches is enough to map all other points
(*informal* geometric proof based on line preservation)



Projective: # correspondences?

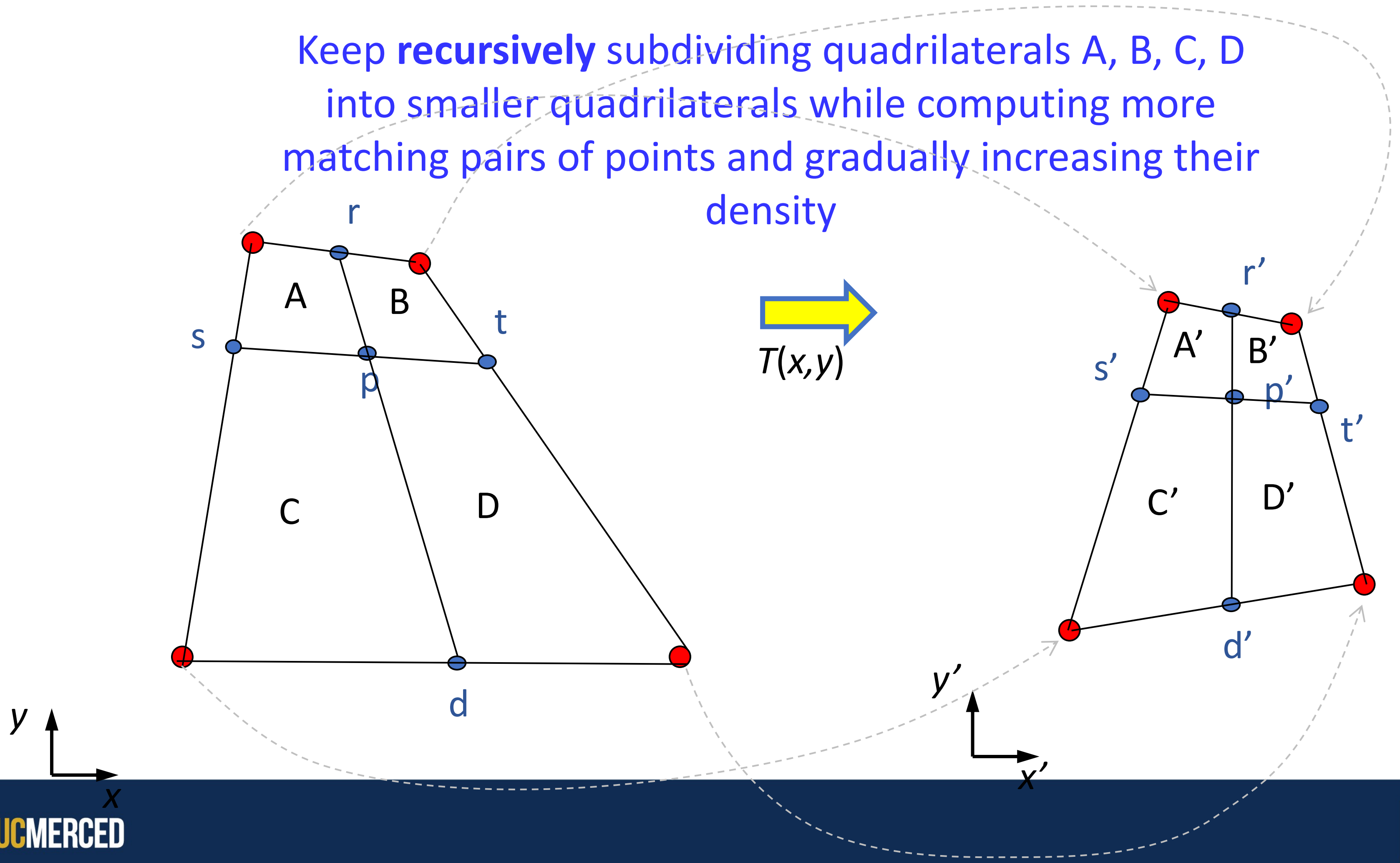
4 matches is enough to map all other points
(*informal* geometric proof based on line preservation)



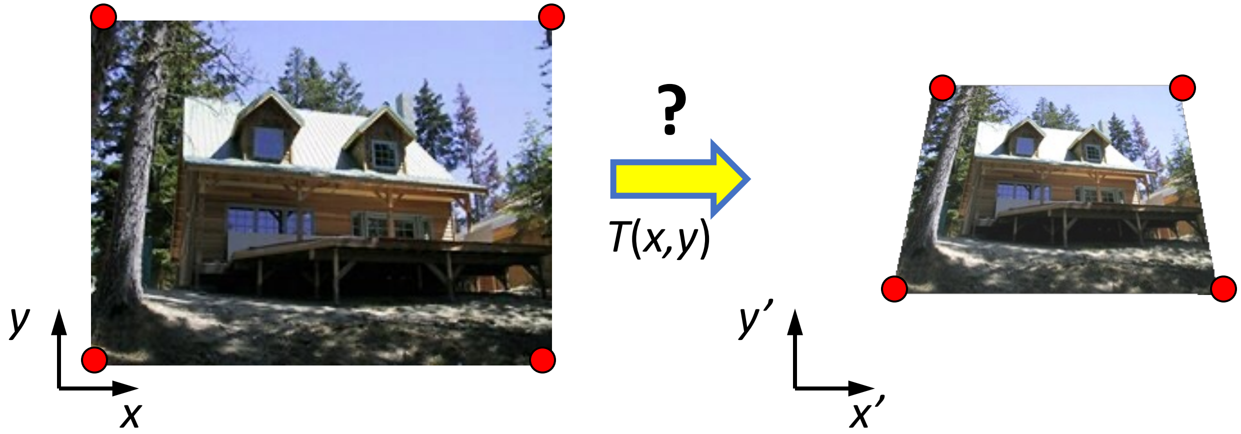
Projective: # correspondences?

4 matches is enough to map all other points
(*informal* geometric proof based on line preservation)

Keep **recursively** subdividing quadrilaterals A, B, C, D into smaller quadrilaterals while computing more matching pairs of points and gradually increasing their density



Projective: # correspondences?



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

=4

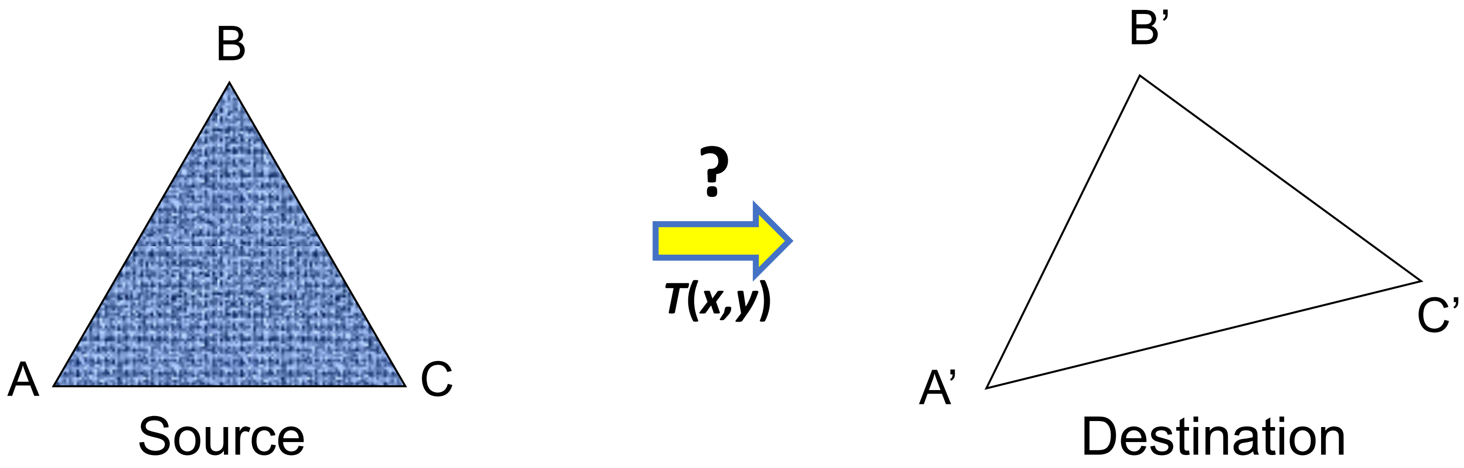
Easy to check that 4 pairs give only 4x2 (=8) equations!
What about 9 unknowns?

$$\mathbf{M} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Homographies have only 8 DOF since scale is irrelevant
(multiplying M by any factor does not change the actual transformation).

More on estimating homographies
from 4 matching pairs of point - later in Topic 5.

Example: warping triangles (e.g. in a mesh)



- Given two triangles: ABC and A’B’C’ in 2D (3 corresponding pairs)
- Need to find a **simple parametric transform T** to transfer all pixels from one to the other ?

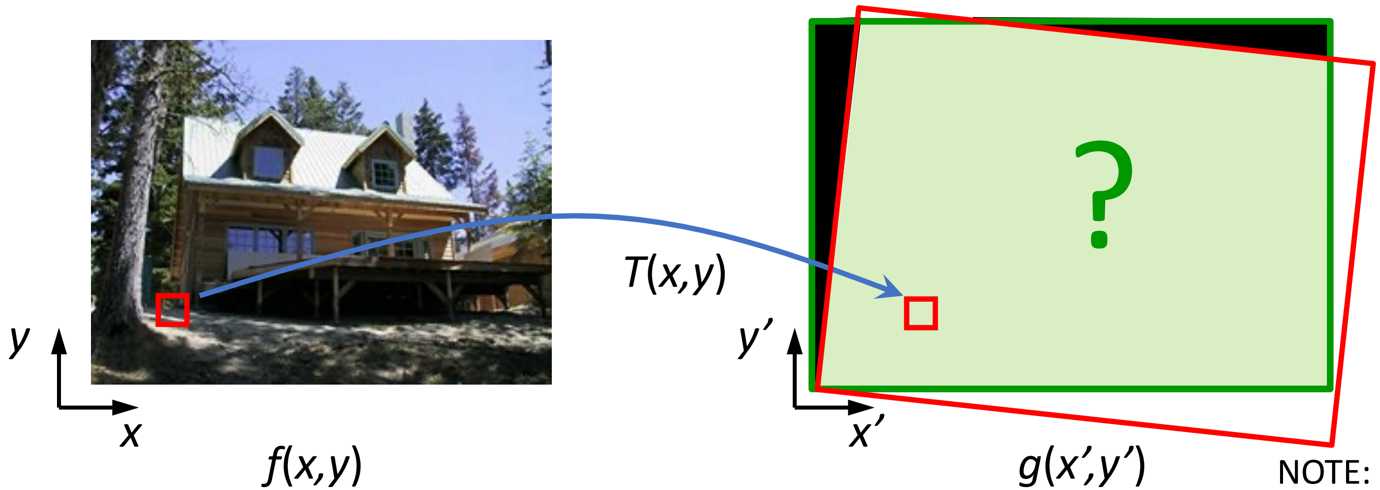
- Common answer: **affine**

$$\mathbf{M} = \begin{bmatrix} a & b & c \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

(solve 6 linear equations with 6 unknowns)

Image warping

assume a given transform T, e.g. rotation or projection



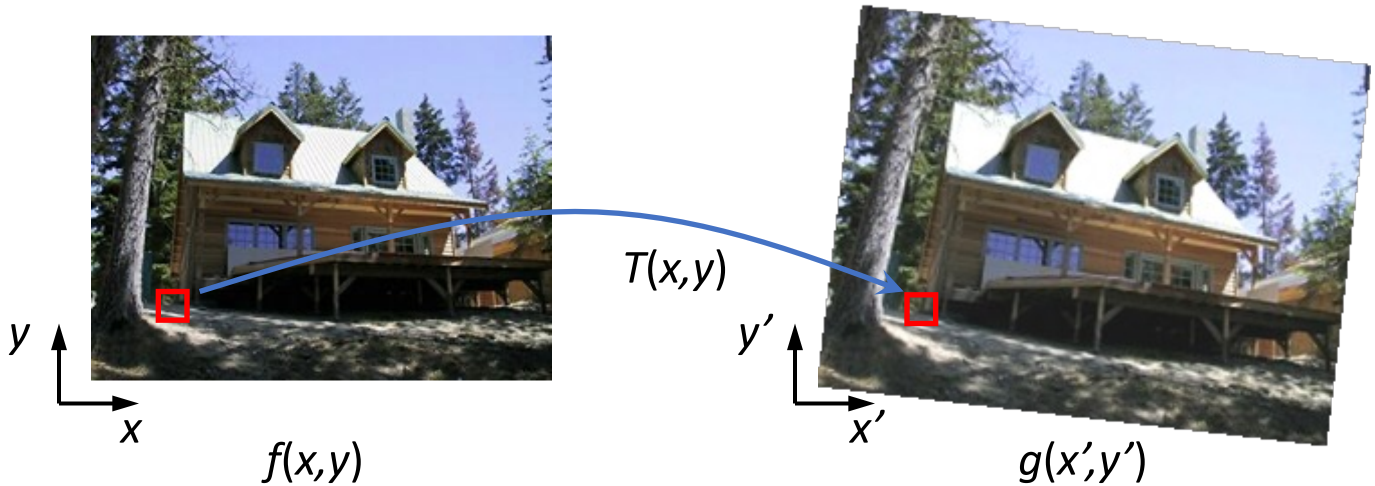
NOTE: in practice, one should consider the **canvas bounds** for the new image

How to generate the transformed image *g* ?

e.g.
- panorama stitching (next topic)
- texture mapping (3D reconstruction)
- novel view generation (special effects, virtual/augmented reality)
- data augmentation (network training)

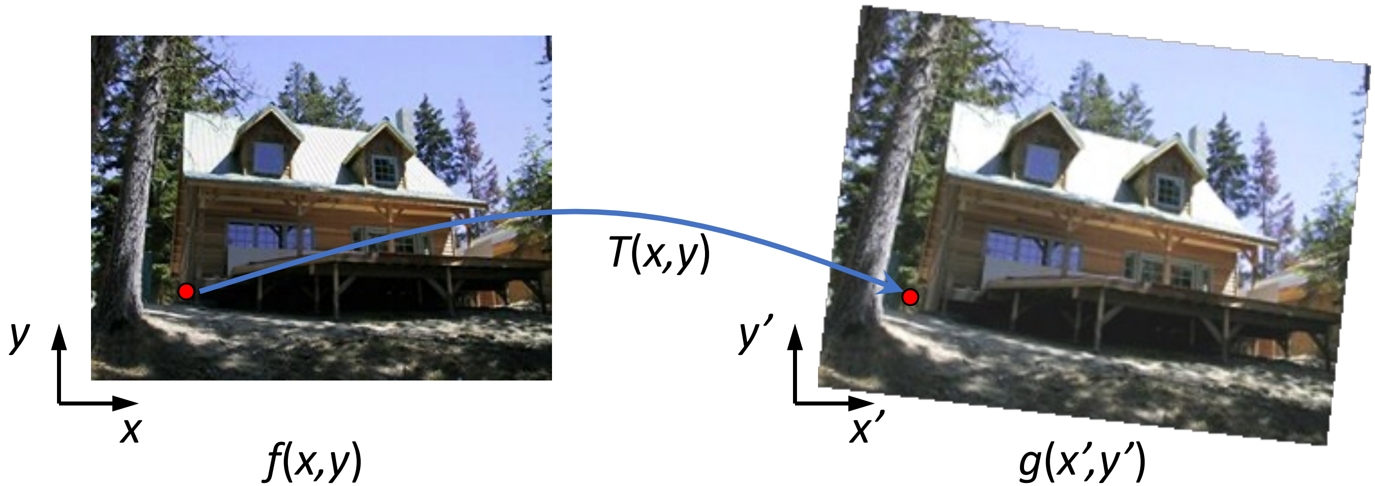
Image warping

COMMENT: for simplicity, the slides ignore the bounds of the new image's canvas, but in your assignments you can not.



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

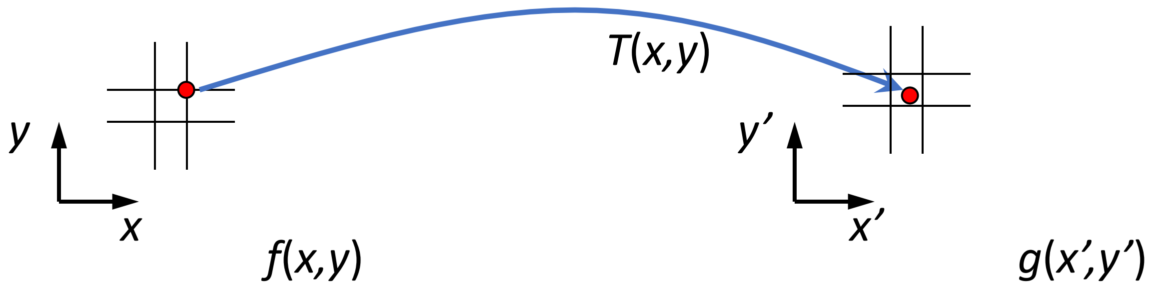
Forward warping



- Send each pixel (x,y) in the first image to its corresponding location $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands "between" two pixels?

Forward warping



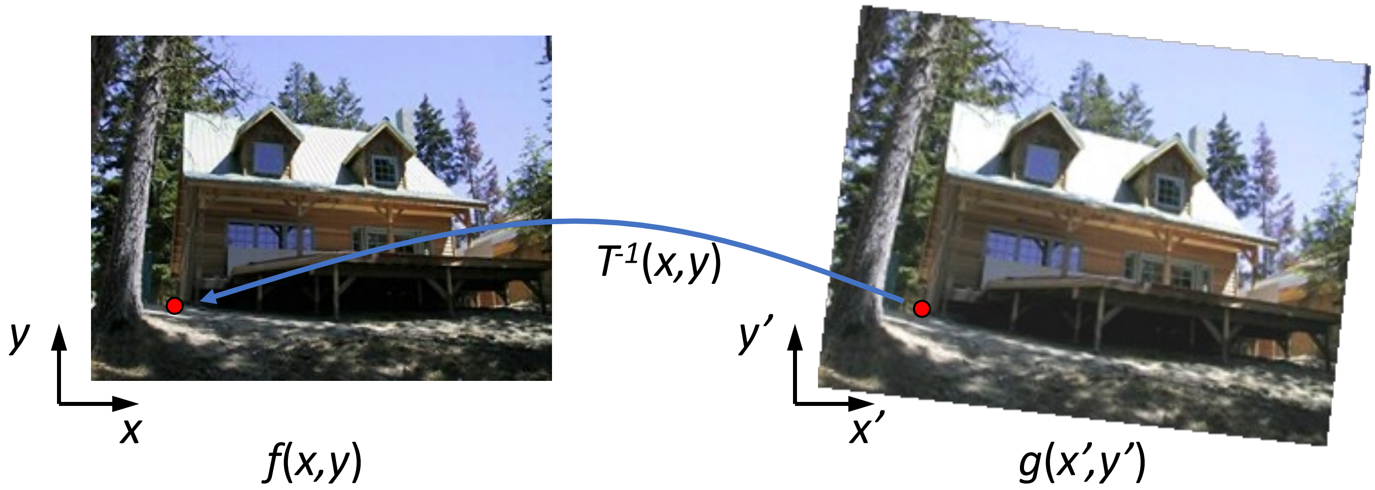
- ❑Send each pixel (*x,y*) in the first image to its corresponding location (*x',y')* = *T*(*x,y*) in the second image

Q: what if pixel lands “between” two pixels?

A: distribute color among neighboring pixels (x’,y’)

- Known as “splatting”

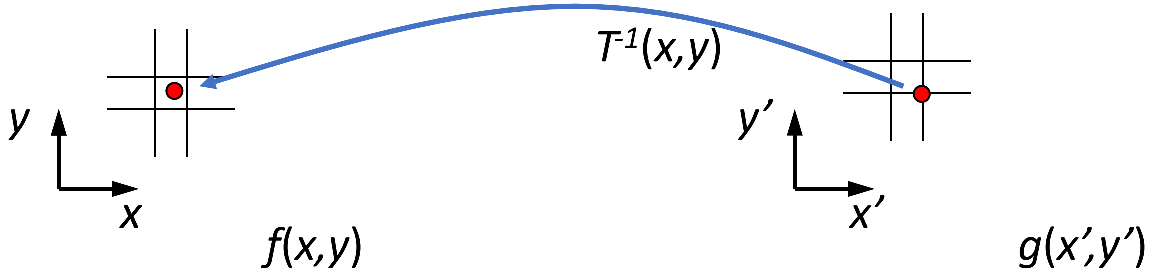
Inverse warping



- Get each pixel (x',y') in the second image from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

Inverse warping



- Get each pixel (x',y') in the second image from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

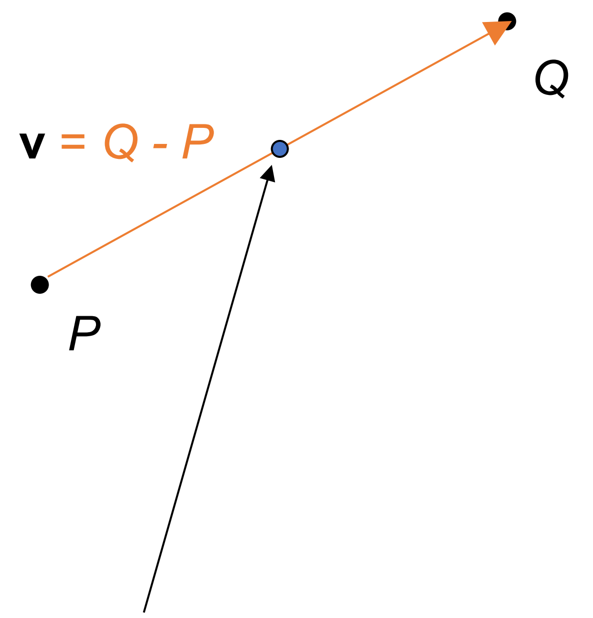
A: *Interpolate* color value from neighbors

- nearest neighbor, bilinear, Gaussian, bicubic

Linear interpolation in vector spaces

Any point between P and Q can be obtained as a linear combination

$$\lambda P + (1-\lambda) Q$$



e.g. $P + 0.5\mathrm{v} = P + 0.5(Q - P)$
$= 0.5P + 0.5\,Q$

NOTE: linear combination

$$\sum \lambda_i V_i \quad \text{for} \quad V_i \in \mathcal{R}^N$$

is called convex combination if

$$\sum_i \lambda_i = 1, \quad \lambda_i \geq 0$$

.

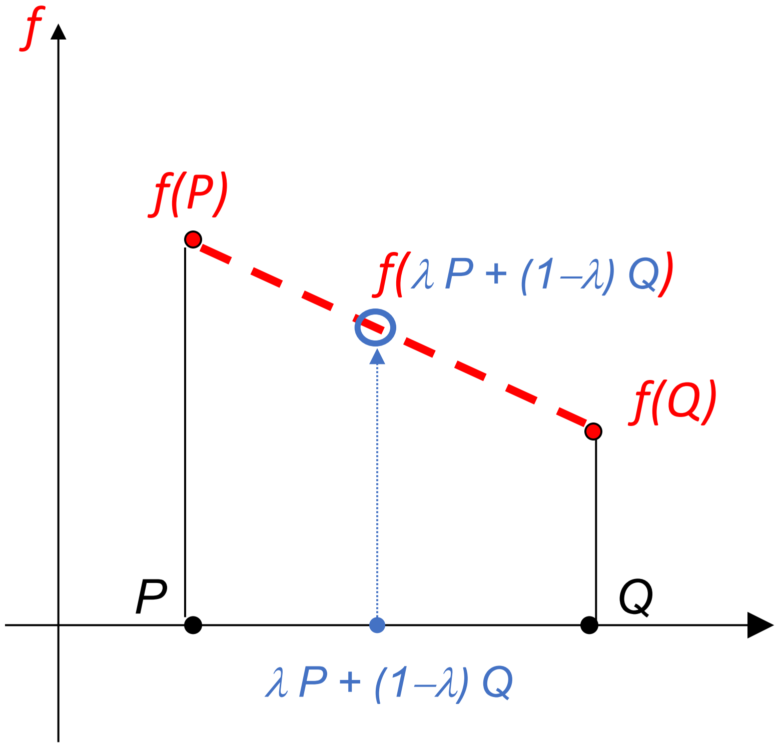
Linear interpolation for functions

Assume 1D image (scan line)
with intensity f(P) and f(Q)
for 2 pixels P and Q

Linear interpolation of function
f between P and Q:

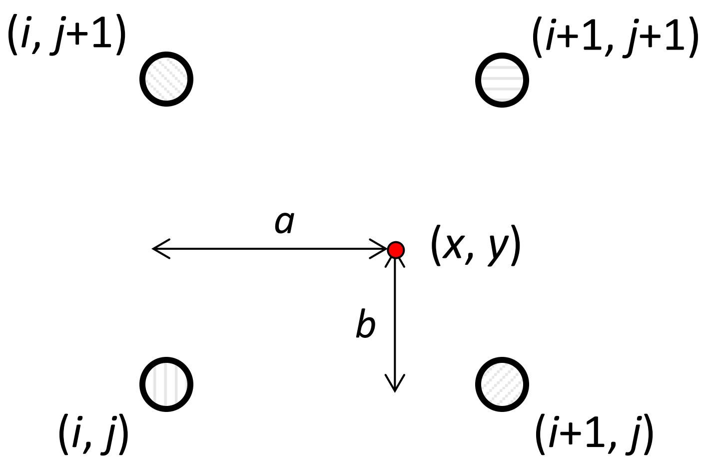
$$f(\lambda P + (1-\lambda) Q) = \lambda f(P) + (1-\lambda) f(Q)$$

In fact, any linear function on [P,Q]
must satisfy the equation above
(by definition of *linear functions*)



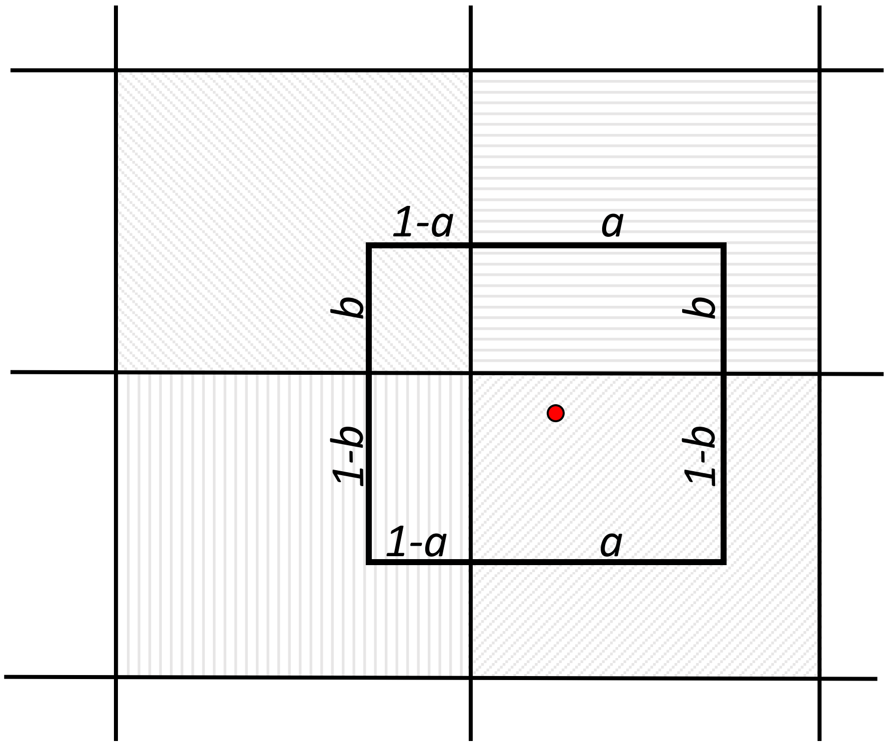
Bilinear interpolation (2 variate image intensity function)

- Sampling of f *at (x,y):*



pixels viewed as points in 2D

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



pixels viewed as square regions in 2D

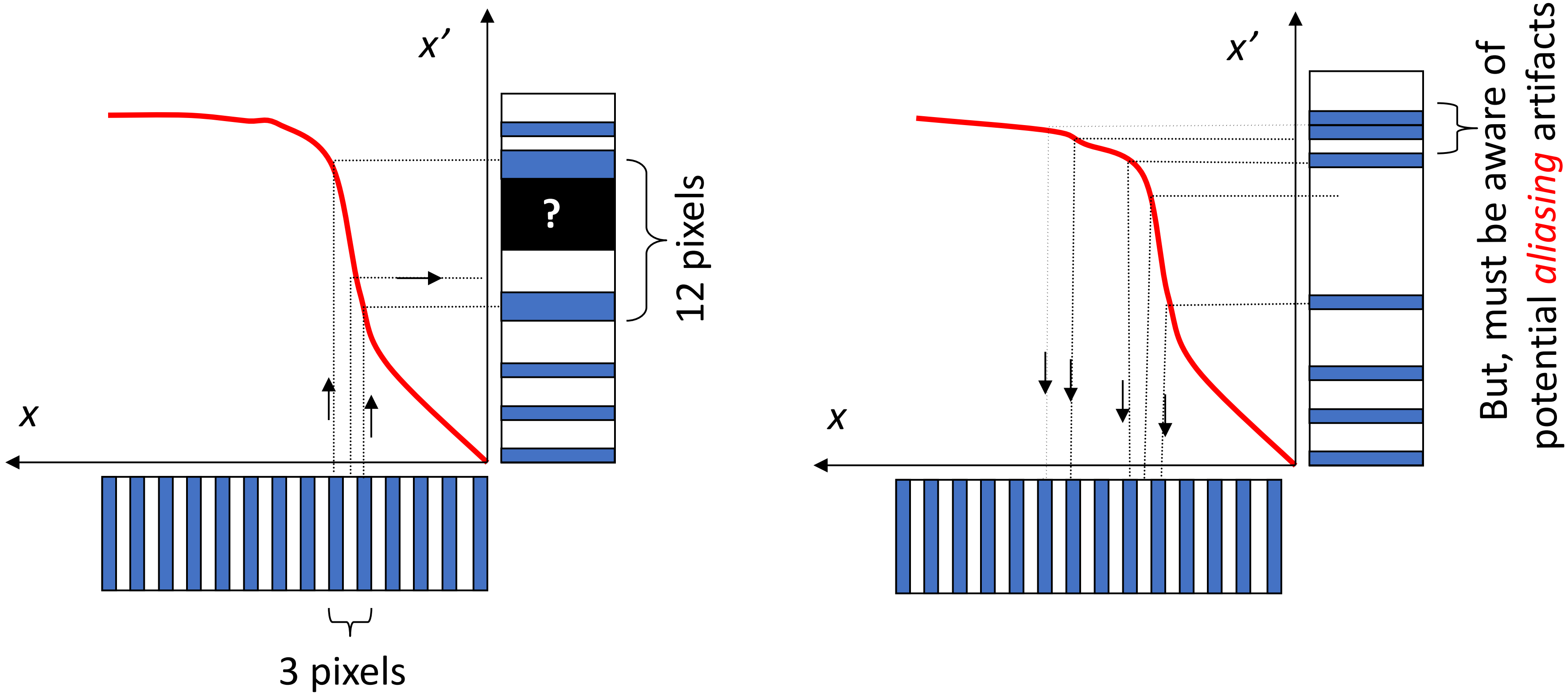
Interpolated intensity at (x,y) can be seen as a weighted average of 4 near-by pixels intensities where weights based on overlap area

Forward vs. inverse warping

- Q: which is better?

A: usually inverse—eliminates holes

- however, it requires an invertible warp function—not always possible...



inverse warping in python

Bug Warning: students often specify the transform from the input image to the output image instead of its inverse

skimage.transform.**warp** (input_image, **inverse_map**,…)

Second argument must be a function transforming coordinates in the output image into their corresponding coordinates in the input image.