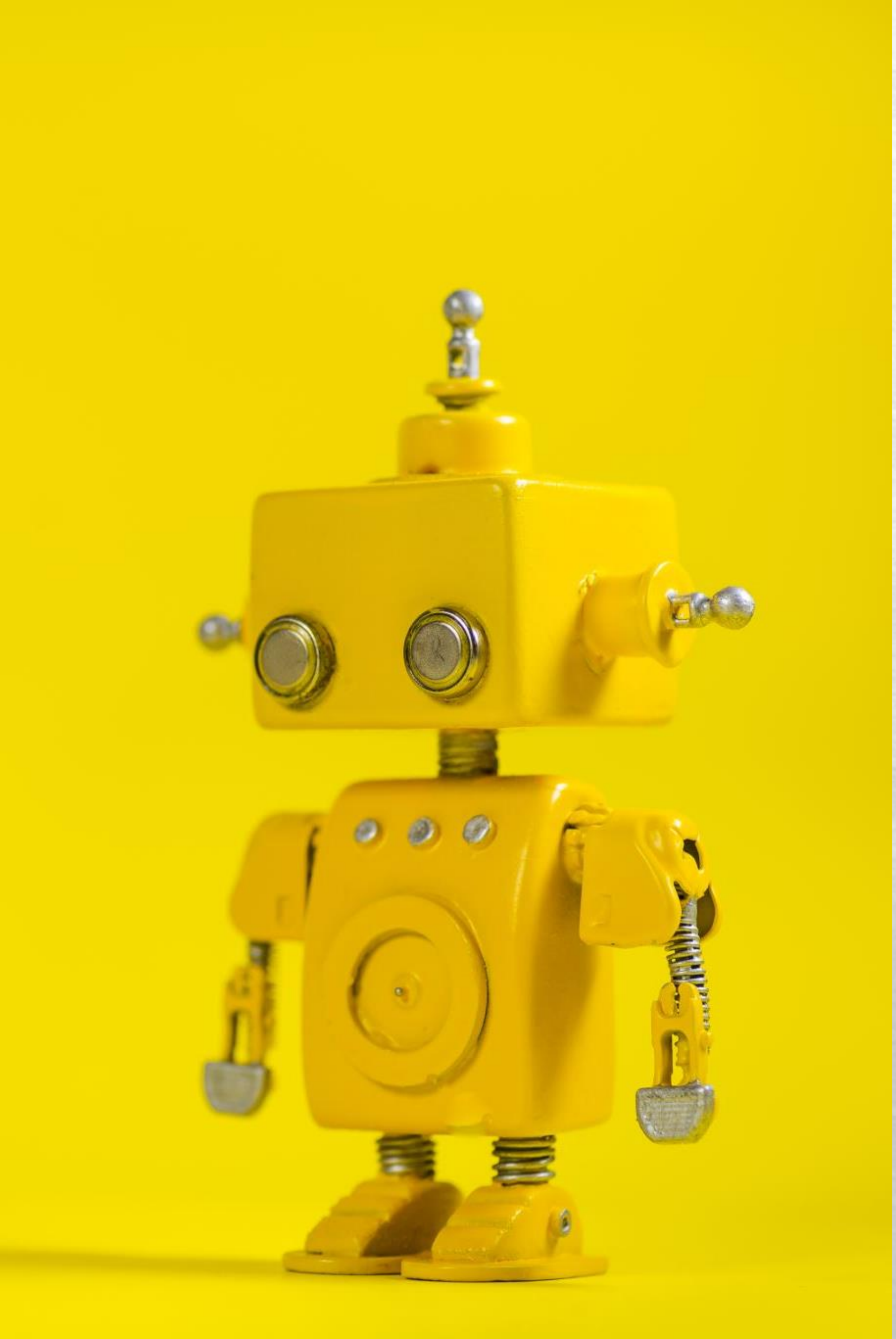


CAP 4453
Robot Vision

Dr. Gonzalo Vaca-Castaño

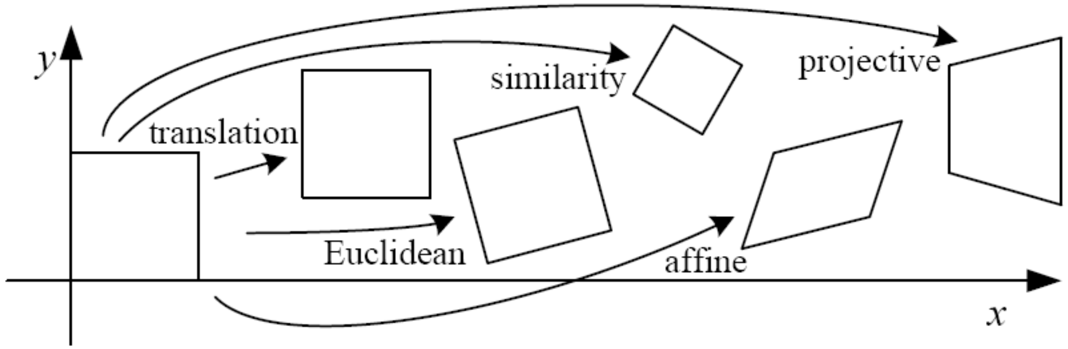
gonzalo.vacacastano@ucf.edu

Credits

- Some slides comes directly from these sources:
  - Ioannis (Yannis) Gkioulekas (CMU)
  - Noah Snavely (Cornell)
  - Marco Zuliani

Short Review from last class

2D image transformations



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

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Projective transformations (aka homographies)

Projective transformations are combinations of

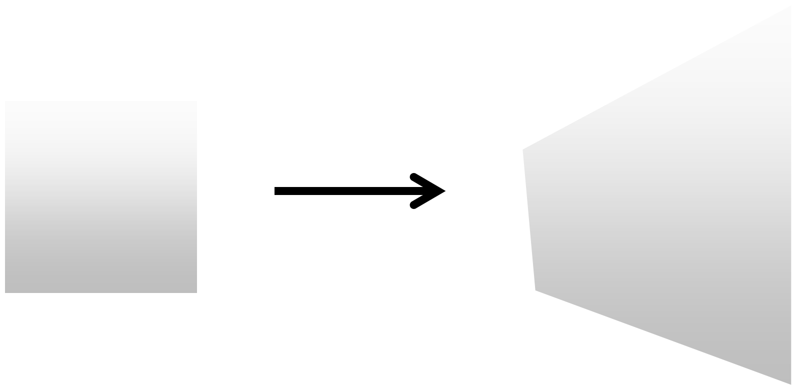
- affine transformations; and
- projective wraps

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

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



Robot Vision

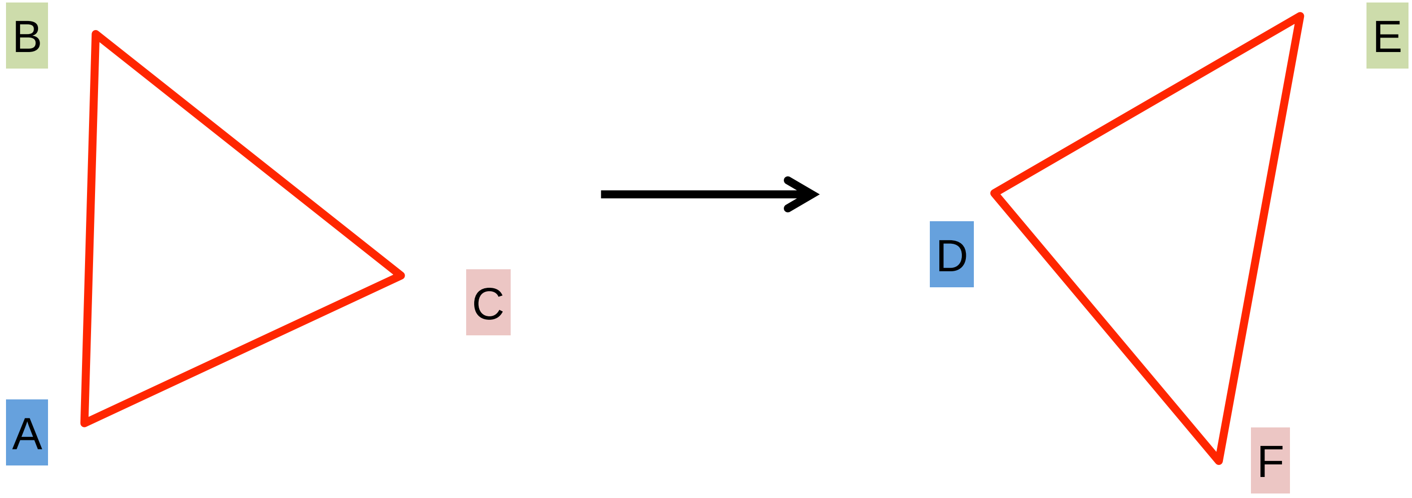
10. Image warping II

Outline

- Linear algebra
- Image transformations
- 2D transformations.
- Projective geometry 101.
- Transformations in projective geometry.
- Classification of 2D transformations.
- **Determining unknown 2D transformations.**
- Determining unknown image warps.

Determining unknown transformations

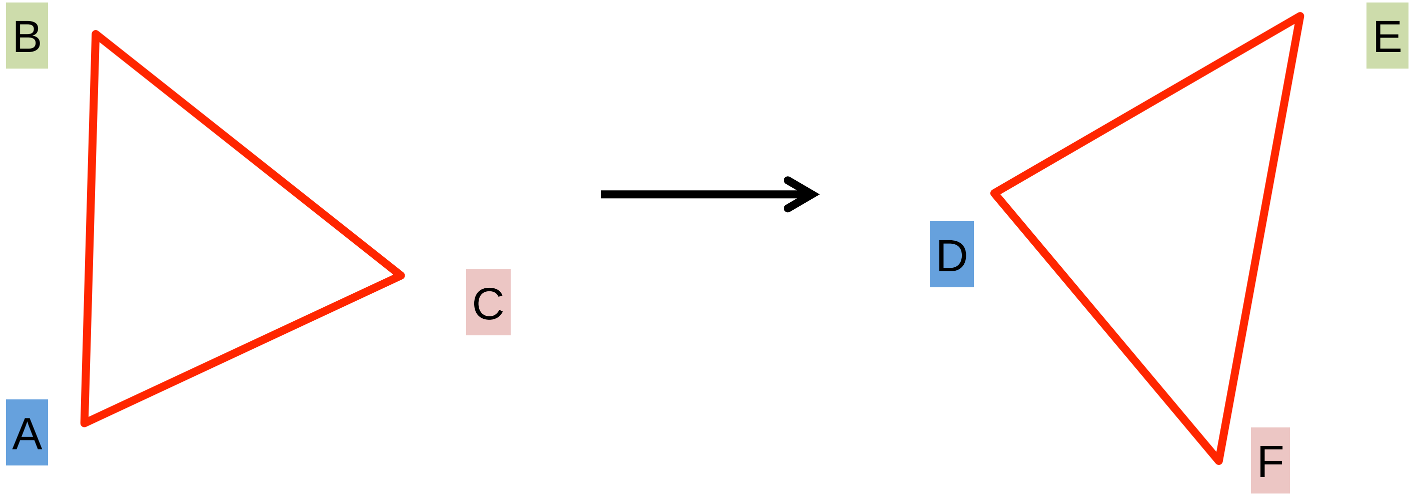
Suppose we have two triangles: ABC and DEF.

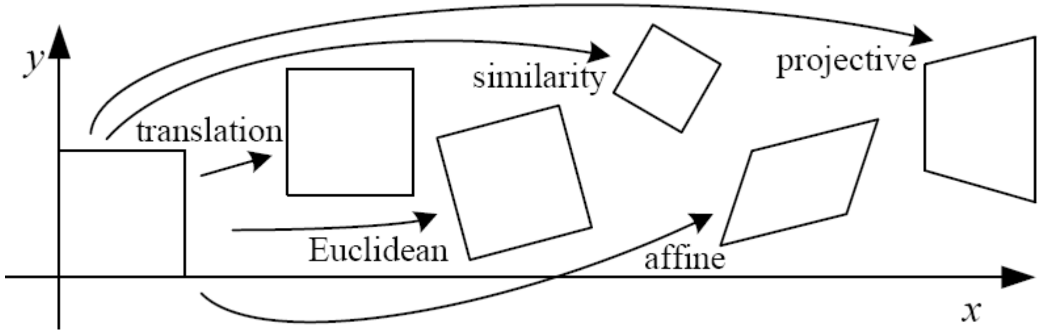


Determining unknown transformations

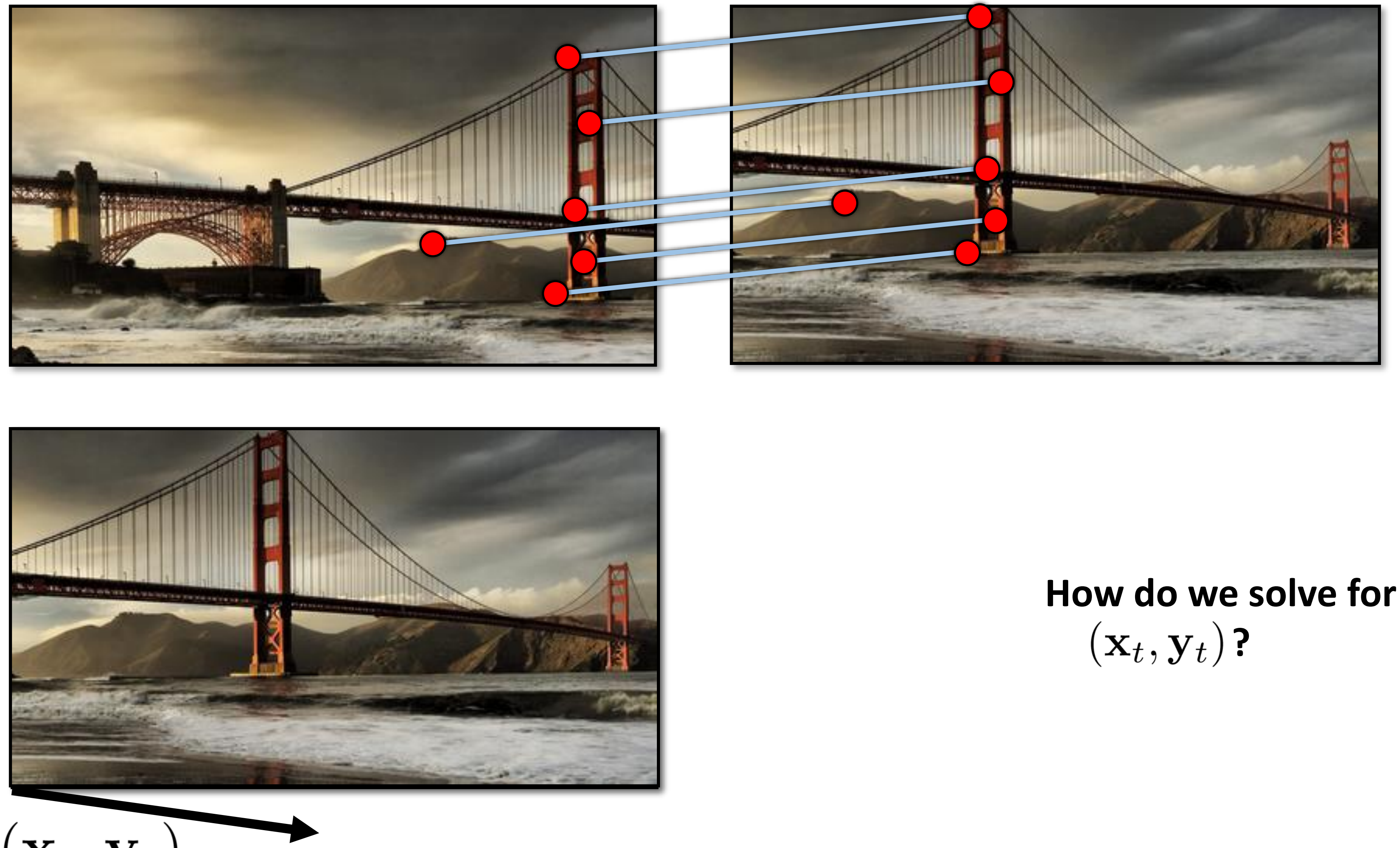
Suppose we have two triangles: ABC and DEF.

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





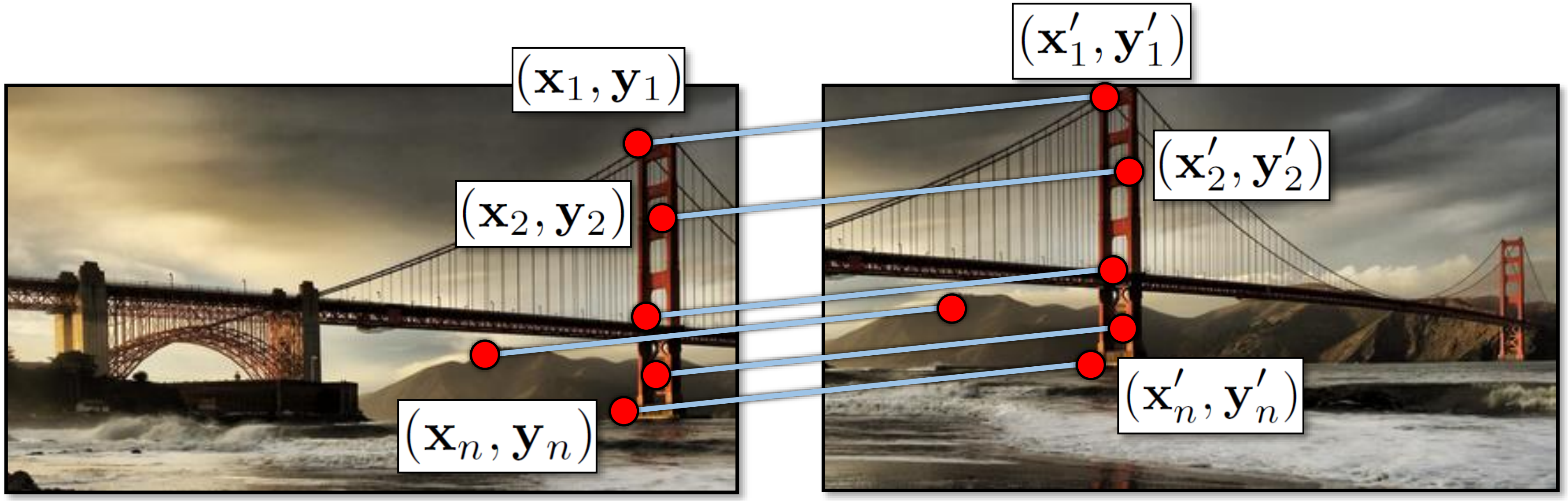
Simple case: translations



$(\mathbf{x}_t, \mathbf{y}_t)$

How do we solve for $(\mathbf{x}_t, \mathbf{y}_t)$ **?**

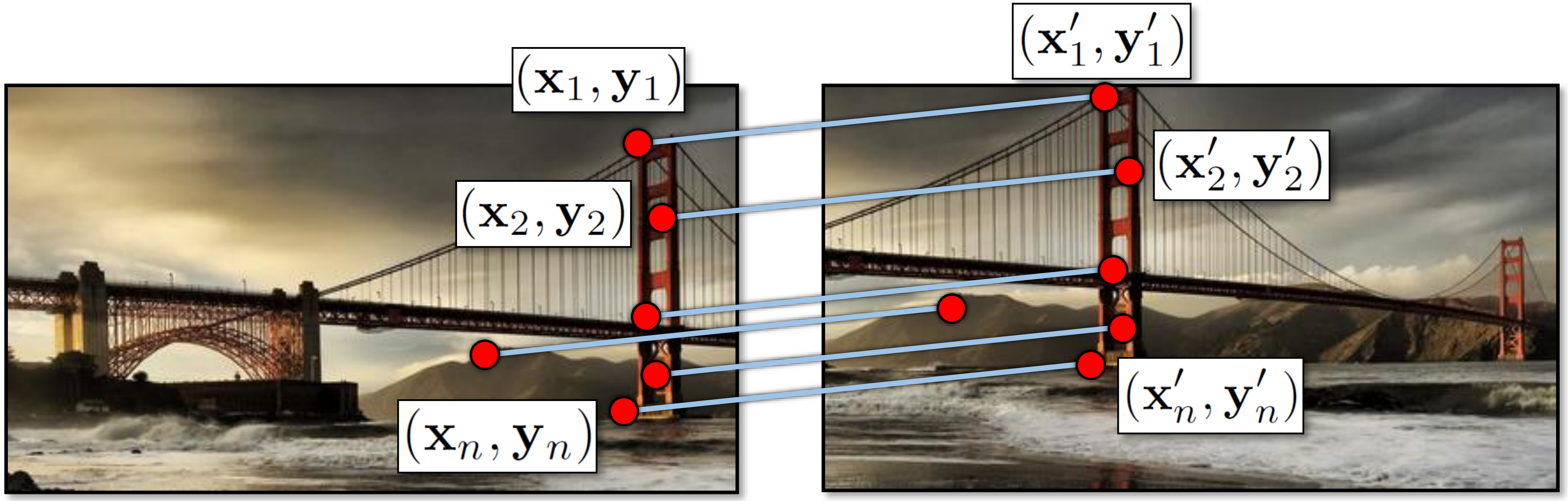
Simple case: translations



Displacement of match *i* = $(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i \right)$$

Another view

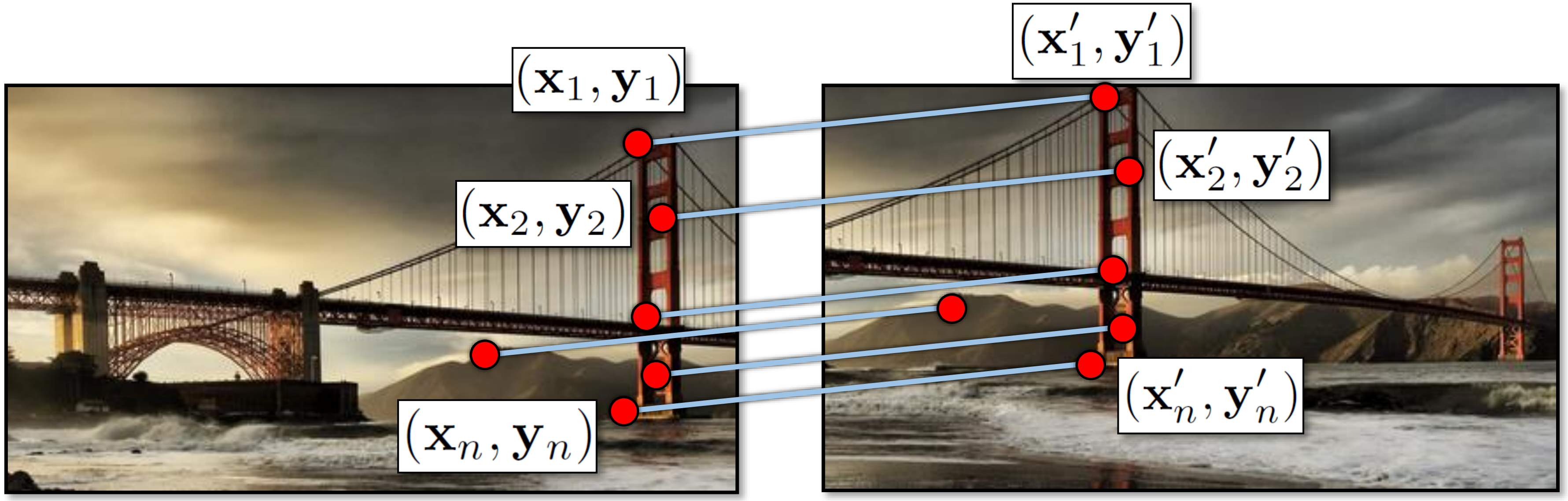


$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

Another view



$$
\begin{aligned}
\mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}'_i \\
\mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}'_i
\end{aligned}
$$

- Problem: more equations than unknowns
  - “Overdetermined” system of equations
  - We will find the *least squares* solution

Least squares formulation

- For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$\begin{aligned} \mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}_i' \\ \mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}_i' \end{aligned}$$

- we define the *residuals* as

$$\begin{aligned} r_{\mathbf{x}_i}(\mathbf{x}_t) &= (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}_i' \\ r_{\mathbf{y}_i}(\mathbf{y}_t) &= (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}_i' \end{aligned}$$

Least squares formulation

- Goal: minimize sum of squared residuals

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^{n} \left(r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2 \right)$$

- "Least squares" solution
- For translations, is equal to mean (average) displacement

Least squares formulation

- Can also write as a matrix equation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ & \vdots & \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \times 2}{\mathbf{A}} \quad \underset{2 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

Least squares

$$\mathbf{At} = \mathbf{b}$$

- Find **t** that minimizes

$$||\mathbf{At} - \mathbf{b}||^2$$

- To solve, form the *normal equations*

$$\mathbf{A}^{\mathrm{T}}\mathbf{At} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

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:

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

Minimize the error:

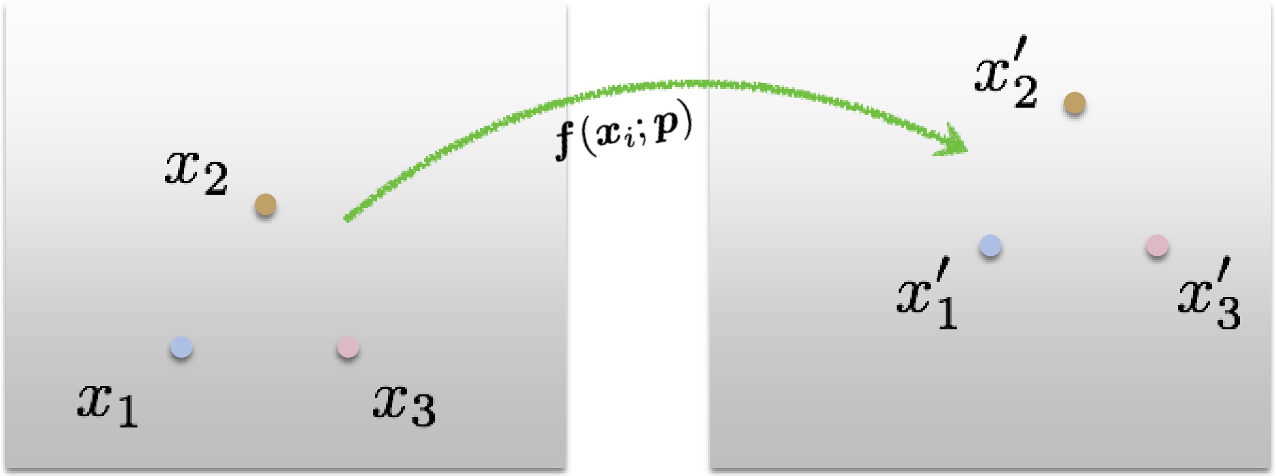
Set derivative to 0 $(\mathbf{A}^\top\mathbf{A})\boldsymbol{x} = \mathbf{A}^\top\boldsymbol{b}$

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

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

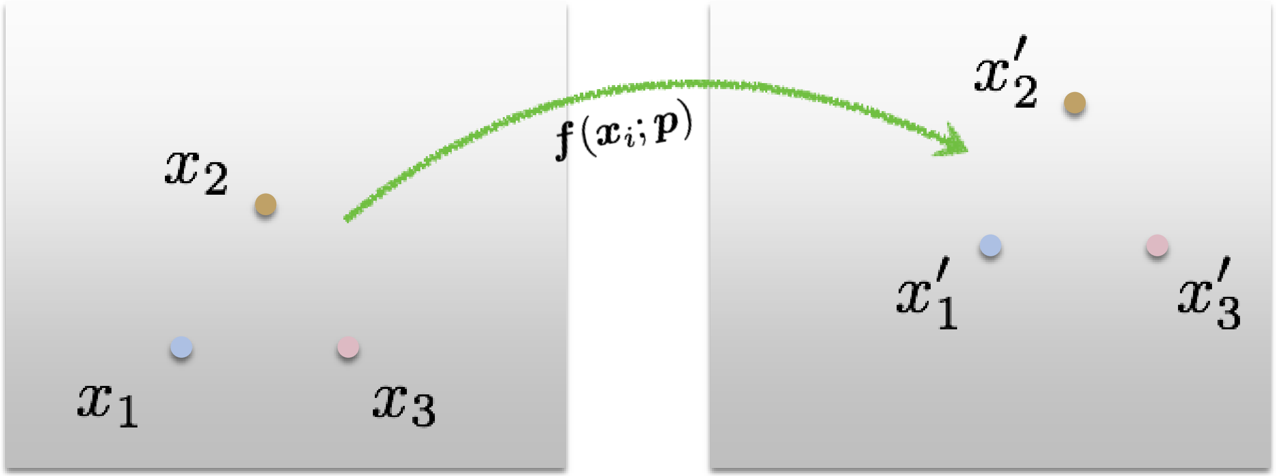
In Phyton:

```python
import numpy as np
x,resid,rank,s = np.linalg.lstsq(A,b)
x
```



Least Squares Error

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



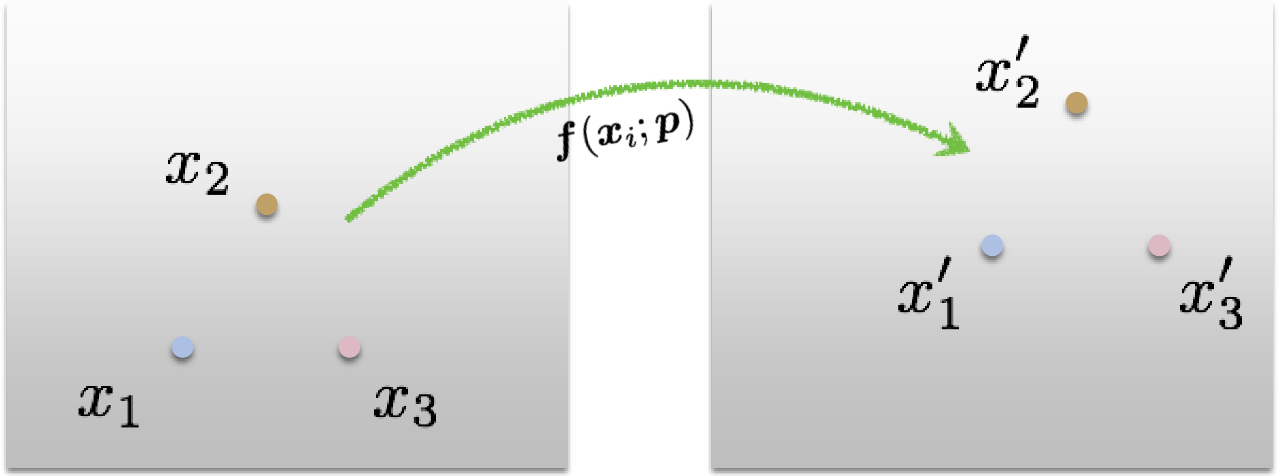
Least Squares Error

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

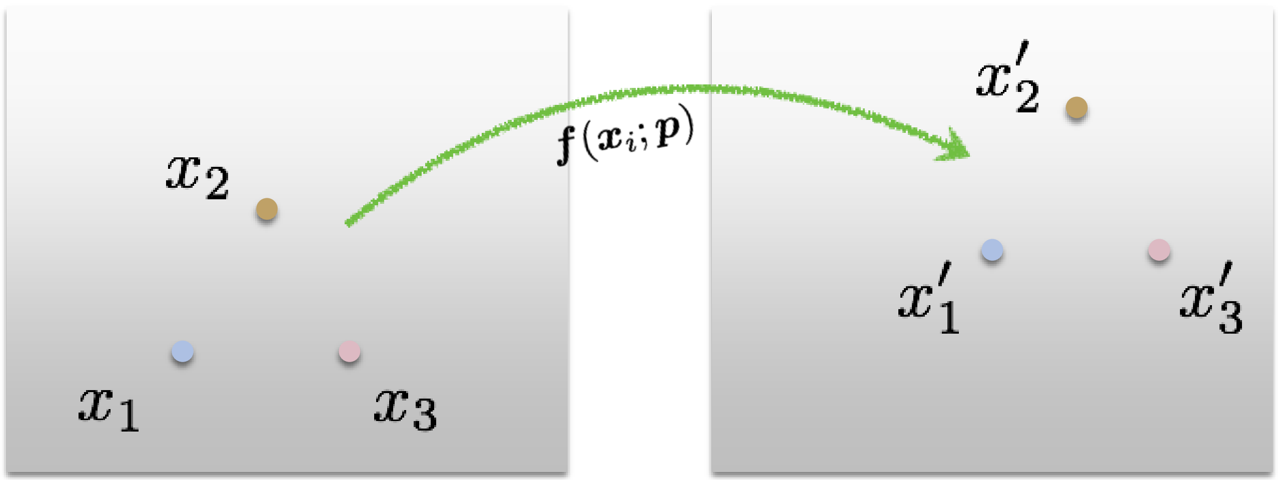
Euclidean (L2) norm

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

squared!

predicted location

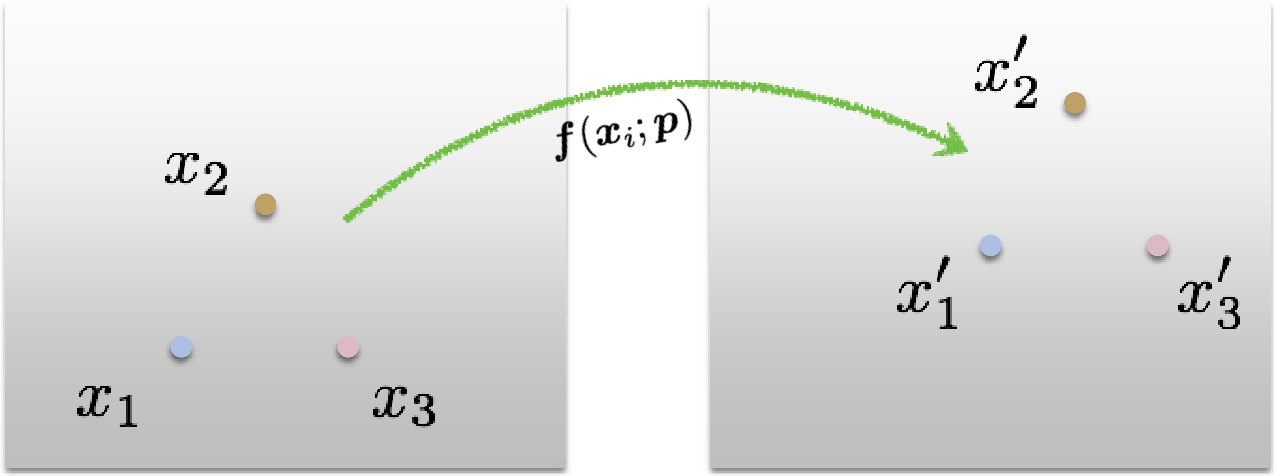
measured location



Least Squares Error

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

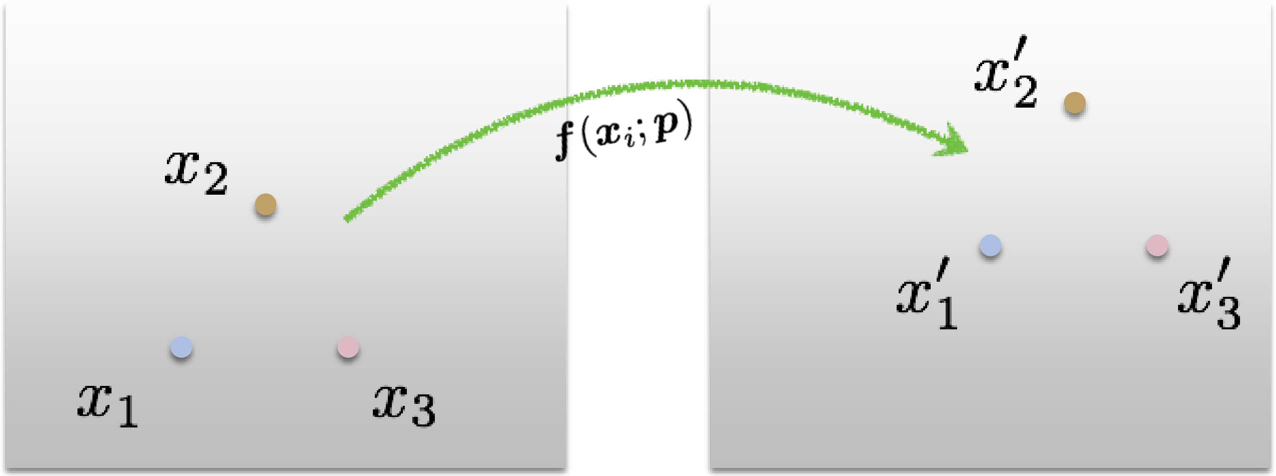
Residual (projection error)



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!)

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

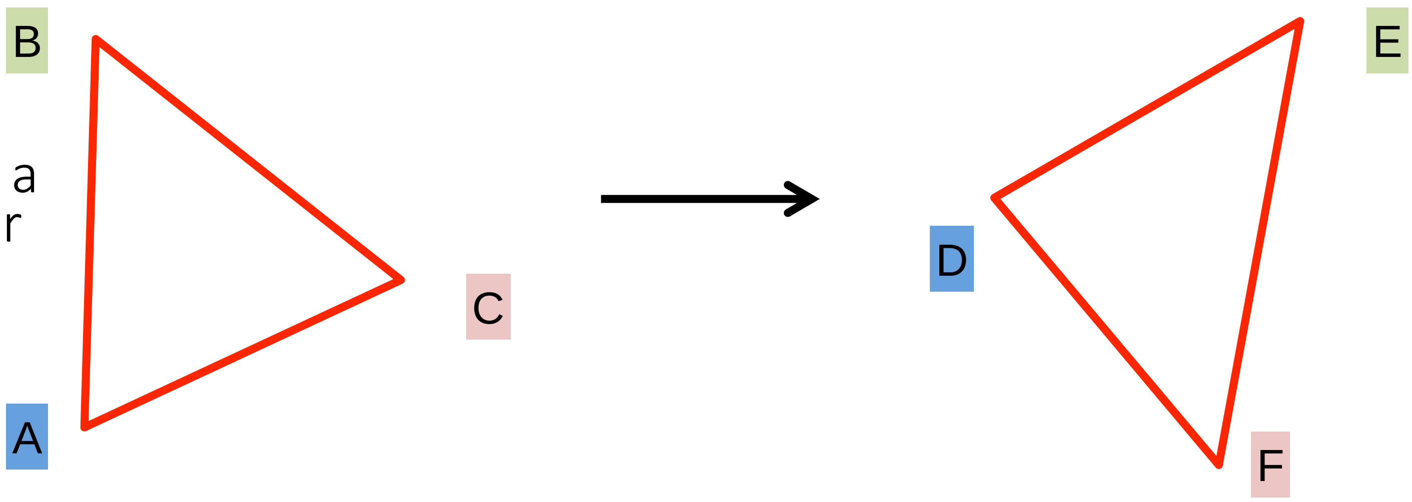
$$= \|\mathbf{A}\boldsymbol{x} - \boldsymbol{b}\|^2 \quad \text{(matrix form)}$$

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?

Important: We will see a different procedure for dealing with homographies!



Affine transform:
uniform scaling + shearing
+ rotation + translation

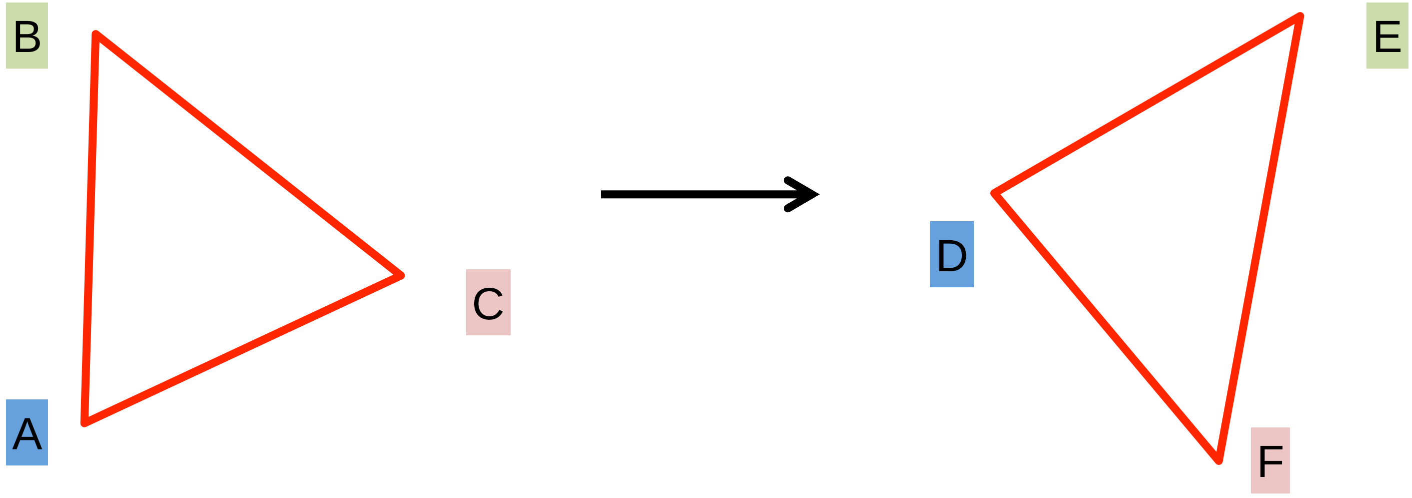
$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

How many degrees of freedom 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

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

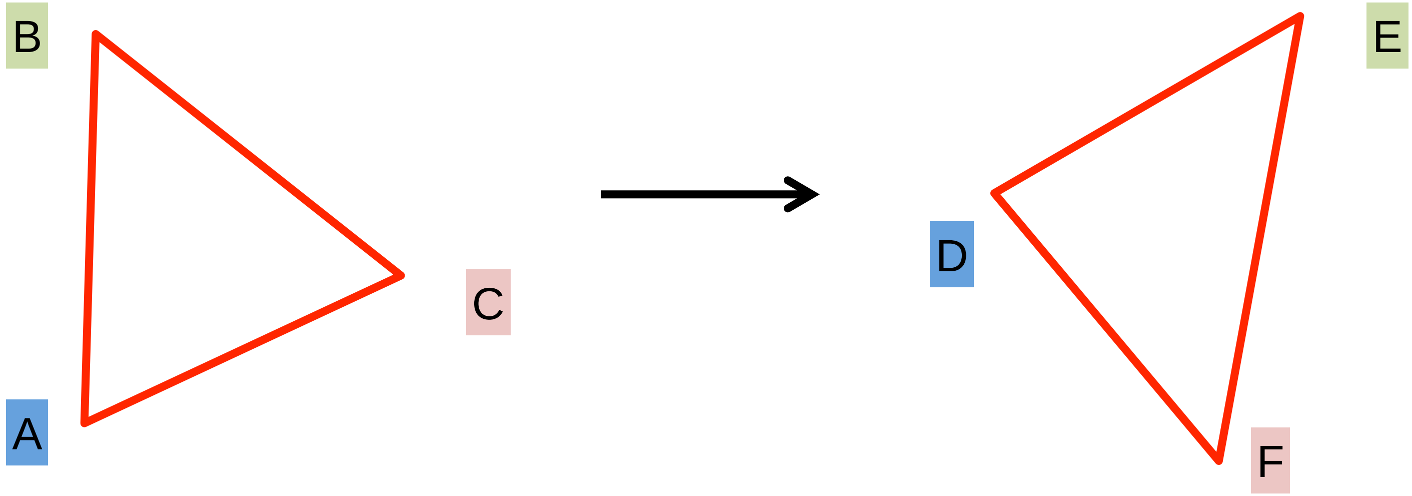
point correspondences

- 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**?

Affine transformations

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

- How many unknowns?
- How many equations per match?
- How many matches do we need?

Affine transformations

- Residuals:

$$
\begin{aligned}
r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\
r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i
\end{aligned}
$$

- Cost function:

$$
C(a,b,c,d,e,f) = \sum_{i=1}^{n} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)
$$

Affine transformations

- Matrix form

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 6}} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}}_{\substack{\mathbf{t} \\ 6 \times 1}} = \underbrace{\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}}_{\substack{\mathbf{b} \\ 2n \times 1}}$$

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:

Notation in system form:

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

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:

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

Minimize the error:

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

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

Note: You almost never want to compute the inverse of a matrix.

In Phyton:

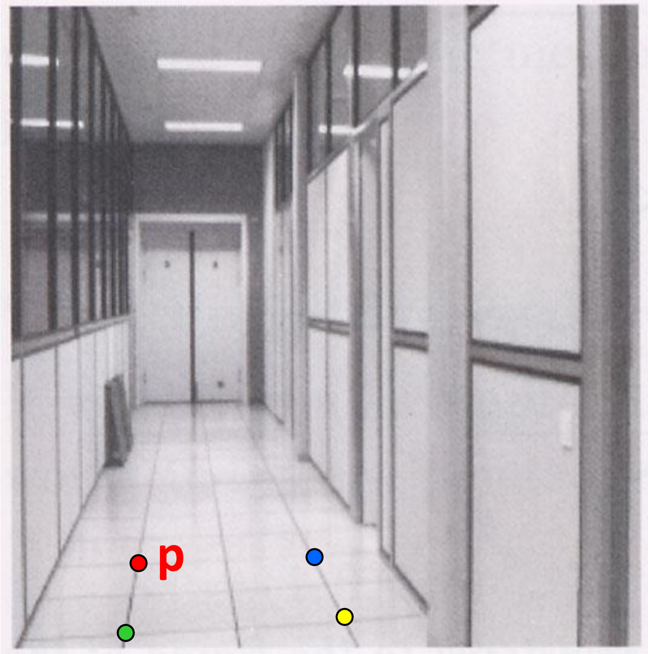
```
import numpy as np
x,resid,rank,s = np.linalg.lstsq(A,b)
x
```

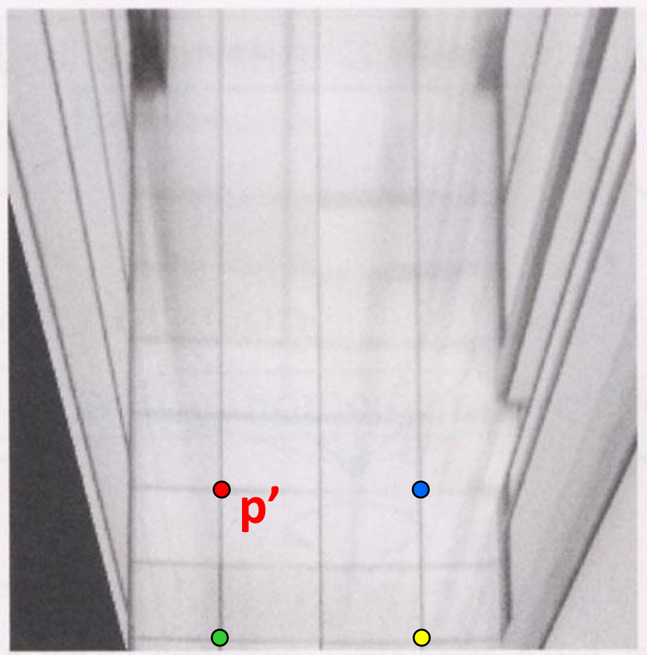
Linear least squares estimation only works when the transform function is ?

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

Also doesn't deal well with outliers (next class !!!)

Homographies





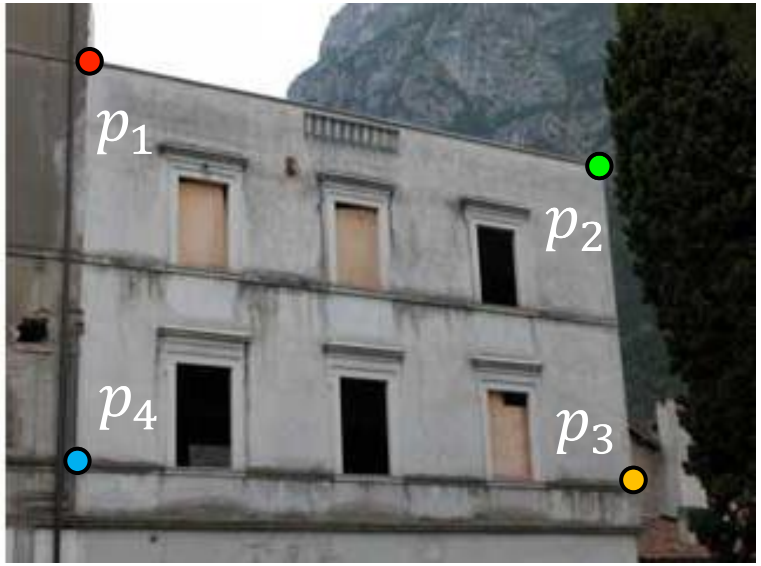
To unwarp (rectify) an image

- solve for homography **H** given **p** and **p'**
- solve equations of the form: w**p'** = **Hp**
  - linear in unknowns: w and coefficients of **H**
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for **H**?

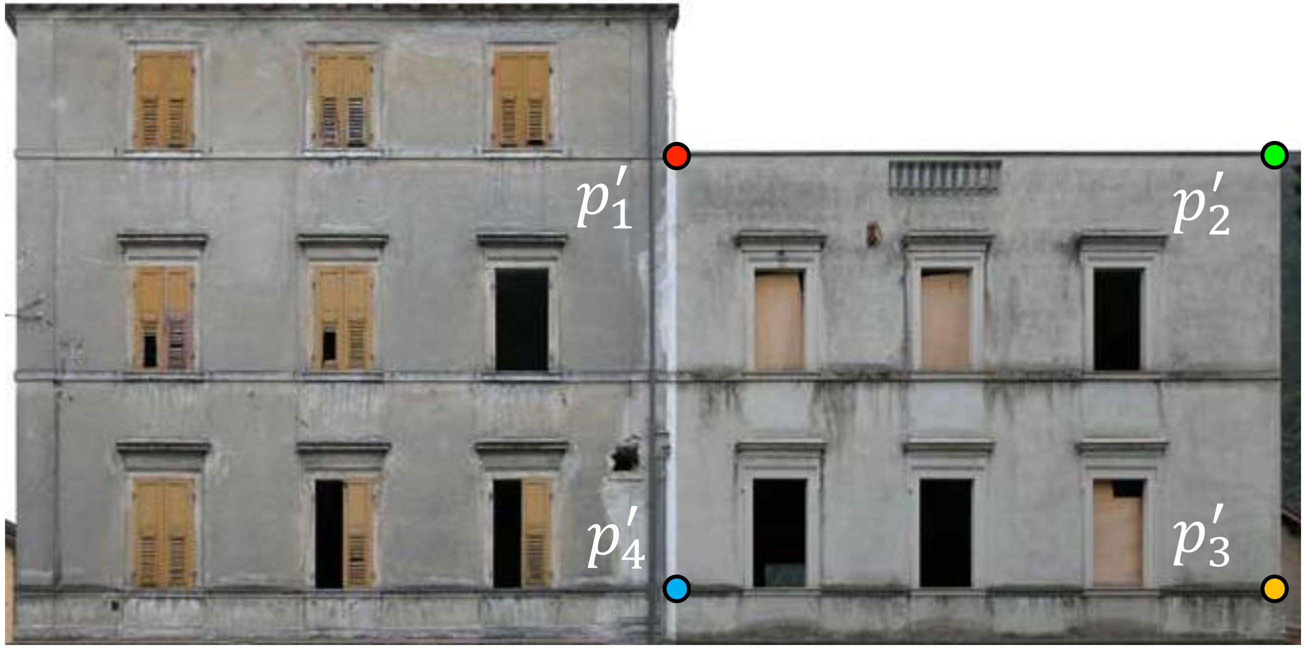
Create point correspondences

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:

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$x' = \alpha(h_1 x + h_2 y + h_3)$$
$$y' = \alpha(h_4 x + h_5 y + h_6)$$
$$1 = \alpha(h_7 x + h_8 y + h_9)$$

Determining the homography matrix

Write out linear equation for each correspondence:

$$P' = H \cdot P \quad \text{or} \quad \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Expand matrix multiplication:

$$x' = \alpha(h_1 x + h_2 y + h_3)$$
$$y' = \alpha(h_4 x + h_5 y + h_6)$$
$$1 = \alpha(h_7 x + h_8 y + h_9)$$

Divide out unknown scale factor:

$$x'(h_7 x + h_8 y + h_9) = (h_1 x + h_2 y + h_3)$$
$$y'(h_7 x + h_8 y + h_9) = (h_4 x + h_5 y + h_6)$$

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

$$h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0$$
$$h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0$$

Solving for homographies

$$
\begin{aligned}
x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\
y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12}
\end{aligned}
$$

$$
\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix}
\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}
= \begin{bmatrix} 0 \\ 0 \end{bmatrix}
$$

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?

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

$$\mathbf{A}_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\boldsymbol{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^\top$$

Solving for homographies

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

Defines a least squares problem: minimize $\|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

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

Reminder: Determining affine transformations

Affine transformation:

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

Vectorize transformation parameters:

Stack equations from point correspondences:

Notation in system form:

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

$$\boxed{Ax = b}$$

Reminder: Determining affine 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 $\boldsymbol{x} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}$ ←

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

In Python:

```python
import numpy as np
x,resid,rank,s = np.linalg.lstsq(A,b)
x
```

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

Singular Value Decomposition

ortho-normal diagonal ortho-normal

unit norm constraint

$$\underset{n \times m}{\mathbf{A}} = \underset{n \times n}{\mathbf{U}}\,\underset{n \times m}{\mathbf{\Sigma}}\,\underset{m \times m}{\mathbf{V}}^\top$$

$$= \sum_{i=1}^{9} \sigma_i \underset{n \times 1}{\boldsymbol{u}_i}\,\underset{1 \times m}{\boldsymbol{v}_i^\top}$$

Singular Value Decomposition

$$A = U \Sigma V^{-1} \qquad \Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \cdot & \\ & & & \sigma_N \end{bmatrix}$$

U, V = orthogonal matrix

$$\sigma_i = \sqrt{\lambda_i}$$

σ = singular value
λ = eigenvalue of $A^t A$

General form of total least squares

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

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

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

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

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

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

minimize (Rayleigh quotient)

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

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

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

Homogeneous Linear Least Squares problem

$$A\mathbf{x} = \mathbf{0}$$

$$A = U\Sigma V^\top = \sum_{i=1}^{9} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$$

- If the homography is *exactly determined*, then $\sigma_9 = 0$, and there exists a homography that fits the points exactly.
- If the homography is *overdetermined*, then $\sigma_9 \geq 0$. Here σ_9 represents a "residual" or goodness of fit.
- We will not handle the case of the homography being *underdetermined*.

Solving for H using DLT

Given $\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$ solve for H such that $\boldsymbol{x}' = \mathbf{H}\boldsymbol{x}$

1. For each correspondence, create 2x9 matrix $\mathbf{A}_i$

2. Concatenate into single 2n x 9 matrix $\mathbf{A}$

3. Compute SVD of $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

4. Store singular vector of the smallest singular value $\boldsymbol{h} = \boldsymbol{v}_{\hat{i}}$

5. Reshape to get $\mathbf{H}$

Recap: Two Common Optimization Problems

Problem statement

$$\text{minimize} \quad \left\| \mathbf{A}\mathbf{x} - \mathbf{b} \right\|^2$$

$$\text{least squares solution to } \mathbf{A}\mathbf{x} = \mathbf{b}$$

Solution

$$\mathbf{x} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{b}$$

```
import numpy as np
x,resid,rank,s = np.linalg.lstsq(A,b)
```

Problem statement

$$\text{minimize} \quad \mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x} \quad \text{s.t. } \mathbf{x}^T \mathbf{x} = 1$$

$$\text{non - trivial lsq solution to } \mathbf{A}\mathbf{x} = 0$$

Solution

$$[\mathbf{v}, \lambda] = \mathrm{eig}(\mathbf{A}^T \mathbf{A})$$

$$\lambda_1 < \lambda_{2..n} : \mathbf{x} = \mathbf{v}_1$$

Derivation using Least squares

$$Ah = 0$$

The sum squared error can be written as:

$$\begin{aligned} f(\mathbf{h}) &= \frac{1}{2}(A\mathbf{h}-\mathbf{0})^T(A\mathbf{h}-\mathbf{0}) \\ f(\mathbf{h}) &= \frac{1}{2}(A\mathbf{h})^T(A\mathbf{h}) \\ f(\mathbf{h}) &= \frac{1}{2}\mathbf{h}^T A^T A\mathbf{h}. \end{aligned}$$

Taking the derivative of f with respect to $\mathbf{h}$ and setting the result to zero,

$$\begin{aligned} \frac{d}{d\mathbf{h}}f = 0 &= \frac{1}{2}\left(A^T A + (A^T A)^T\right)\mathbf{h} \\ 0 &= A^T A\mathbf{h}. \end{aligned}$$

h should equal the eigenvector of $B = A^T A$ that has an eigenvalue of zero

$$B\vec{h} = \lambda\vec{h}$$

(or, in the presence of noise the eigenvalue closest to zero)

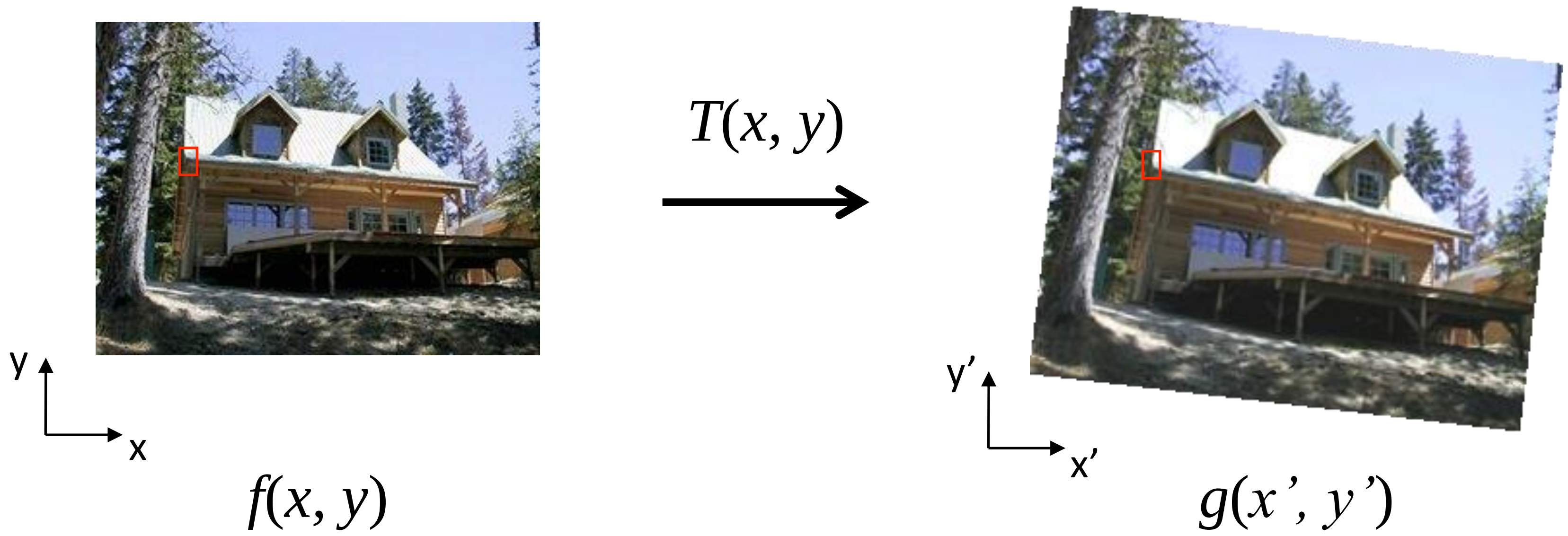
Outline

- Linear algebra
- Image transformations
- 2D transformations.
- Projective geometry 101.
- Transformations in projective geometry.
- Classification of 2D transformations.
- Determining unknown 2D transformations.
- **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?

y

x

$f(x, y)$

$$\xrightarrow{T(x, y)}$$

y'

x'

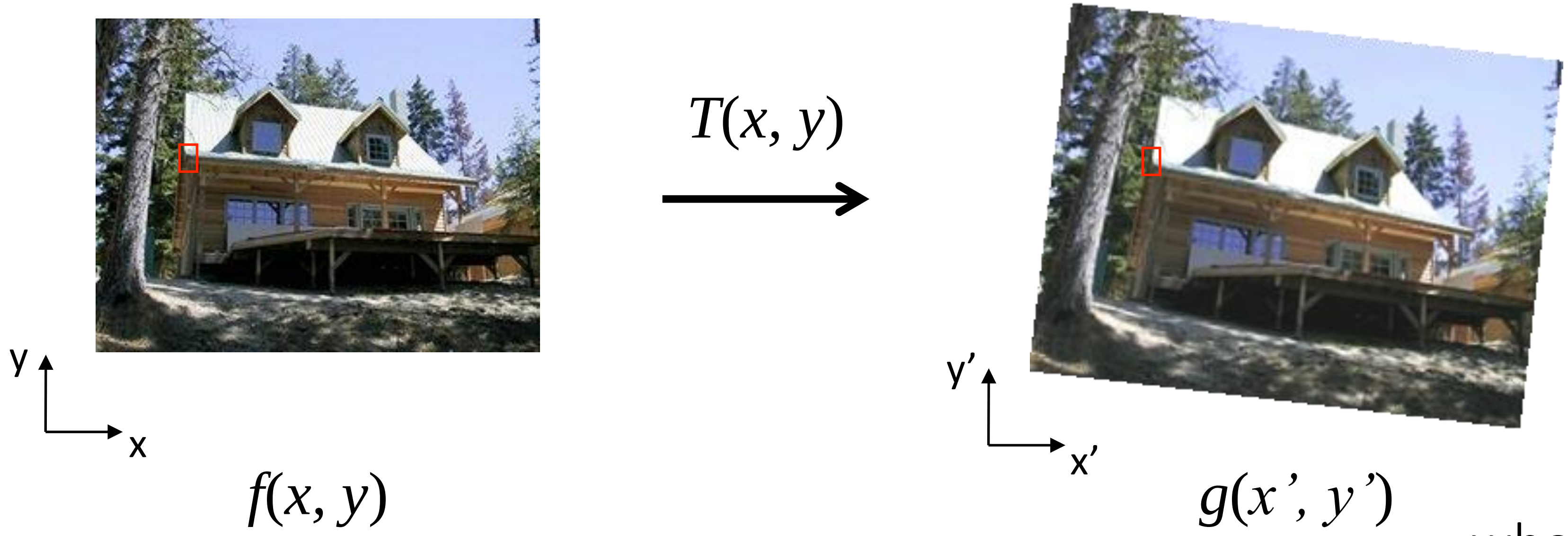
$g(x', y')$

1. Form enough pixel-to-pixel correspondences between two images $\longleftarrow$ later lecture
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?



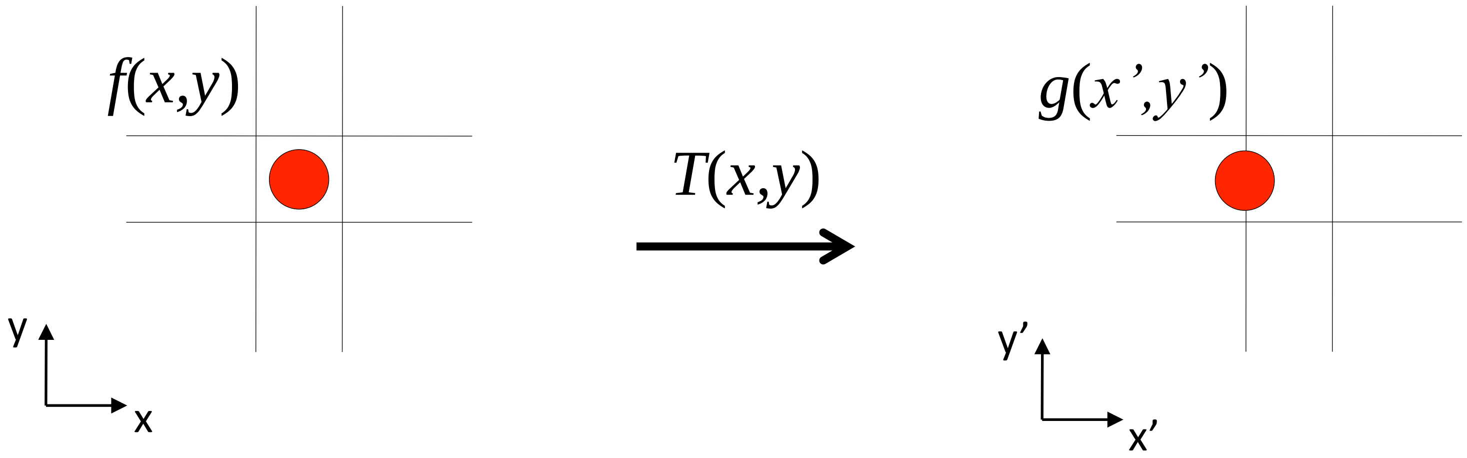
what is the problem with this?

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?

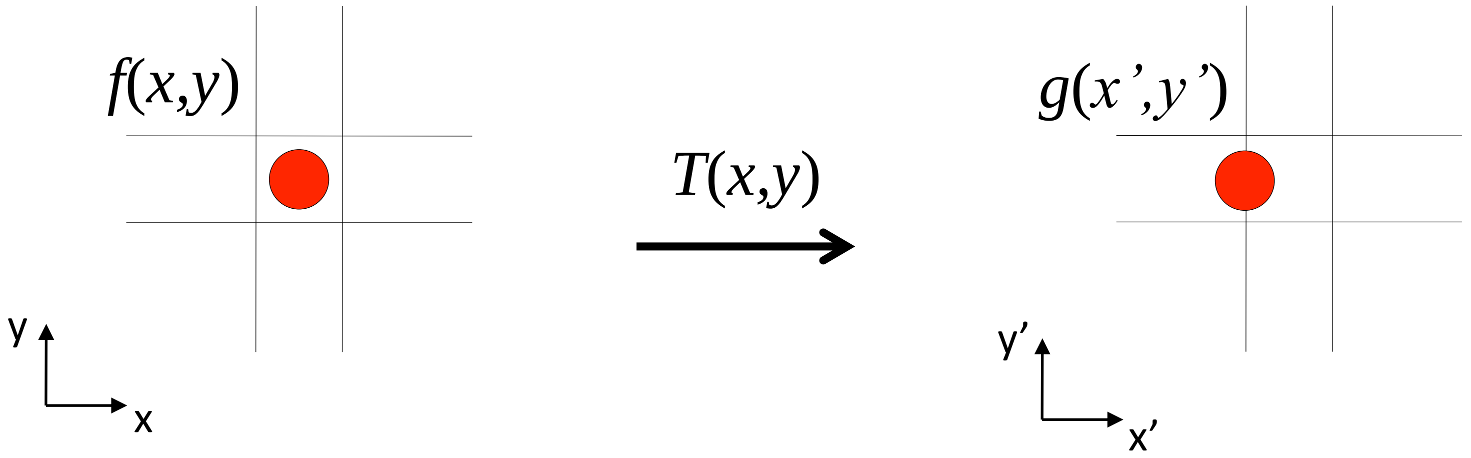


Forward warping

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”)



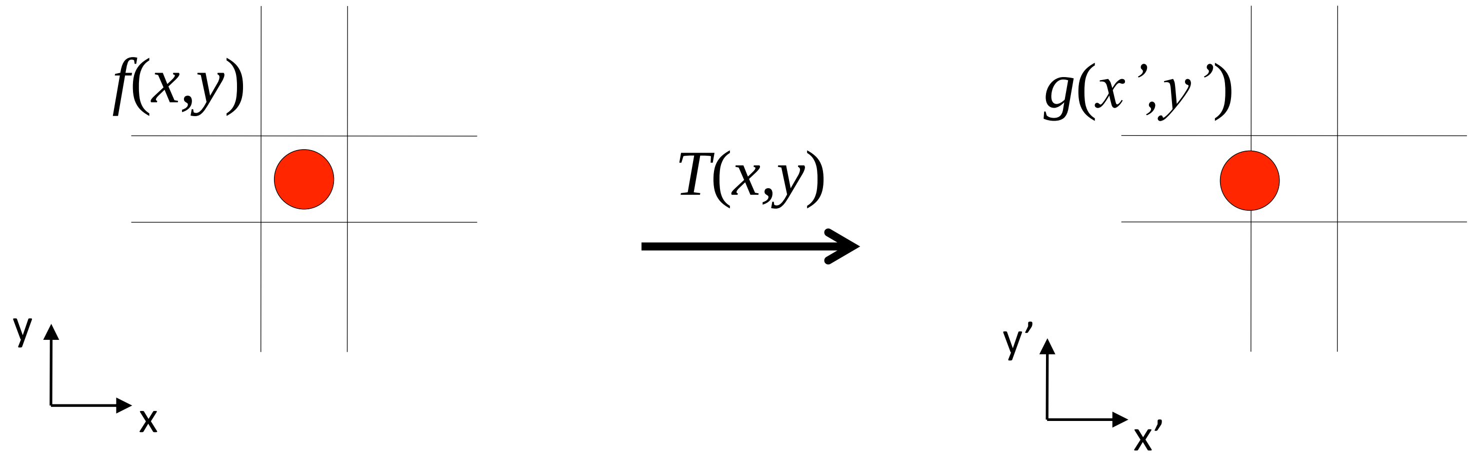
- What if a pixel (x’,y’) receives intensity from more than one pixels (x,y)?

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”)

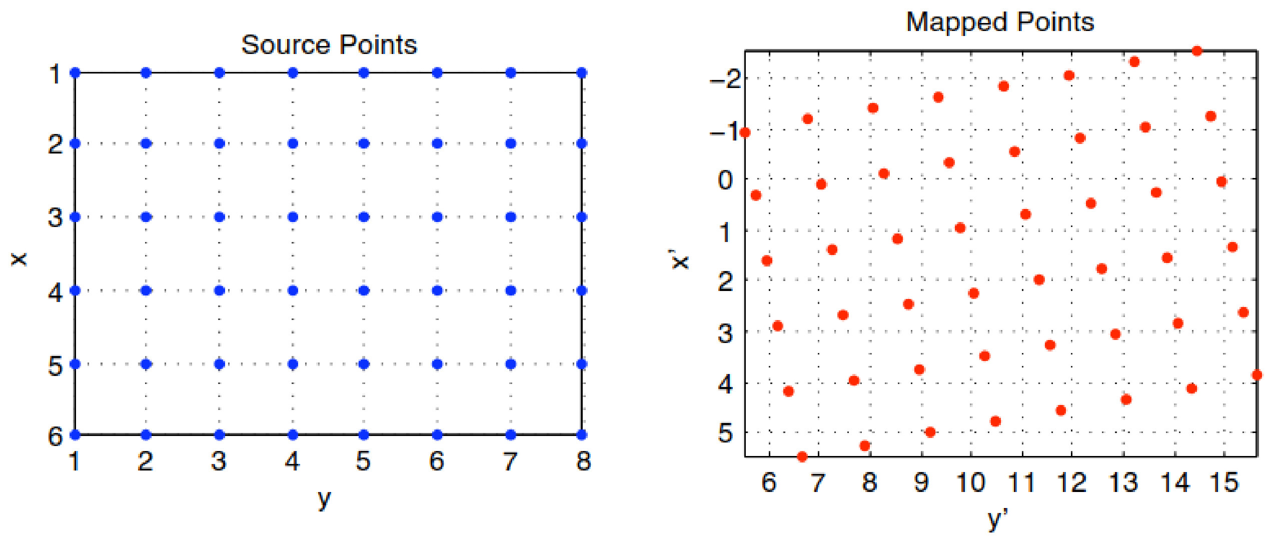


- What if a pixel (x’,y’) receives intensity from more than one pixels (x,y)?

✓ We average their intensity contributions.

Forward mapping example

- Rotation Scale and Translation Mapping

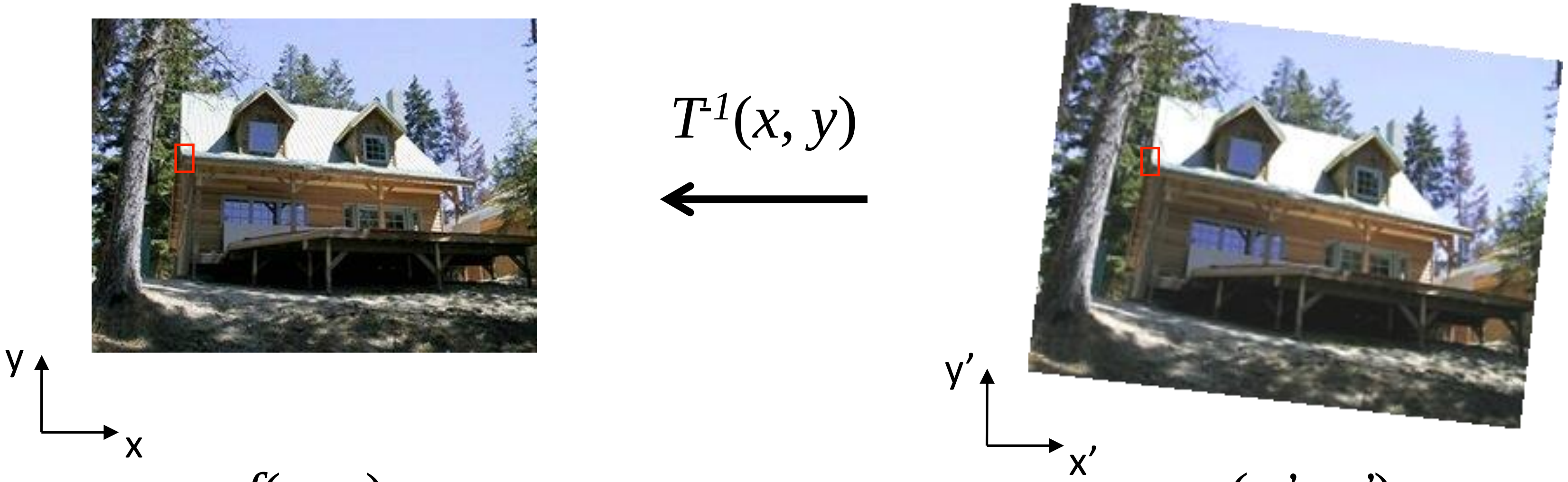


The mapped points do not have integer coordinates!

Inverse warping

Suppose we have two images.

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



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

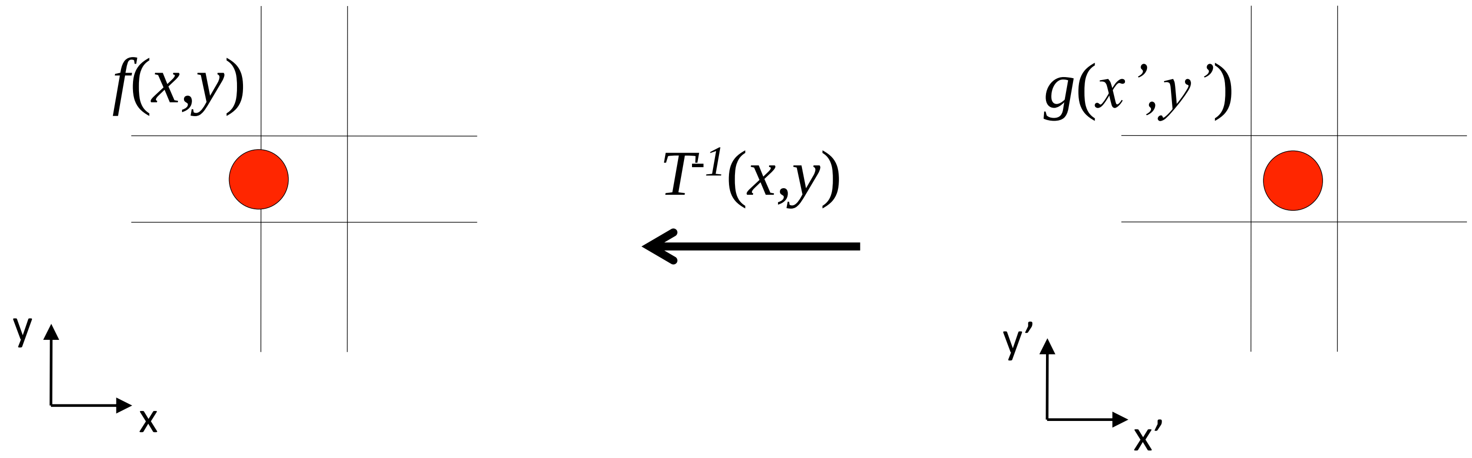
what is the problem with this?

1. Form enough pixel-to-pixel correspondences between two images
2. Solve for linear transform parameters as before, 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?

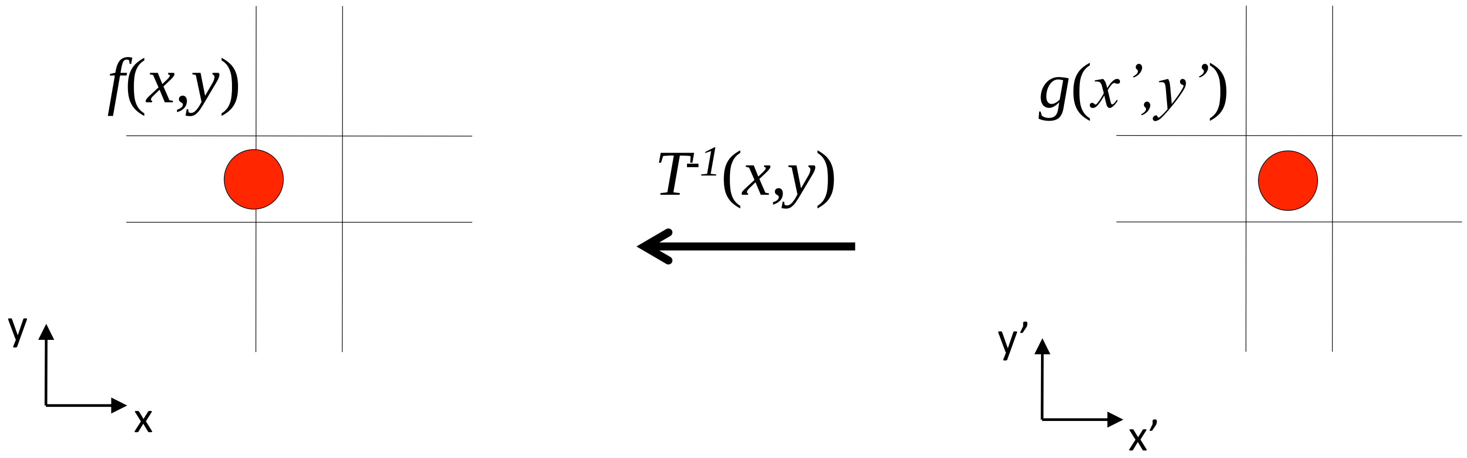


Inverse warping

Pixel may come from between two points

- How do we determine its intensity?

✓ Use interpolation



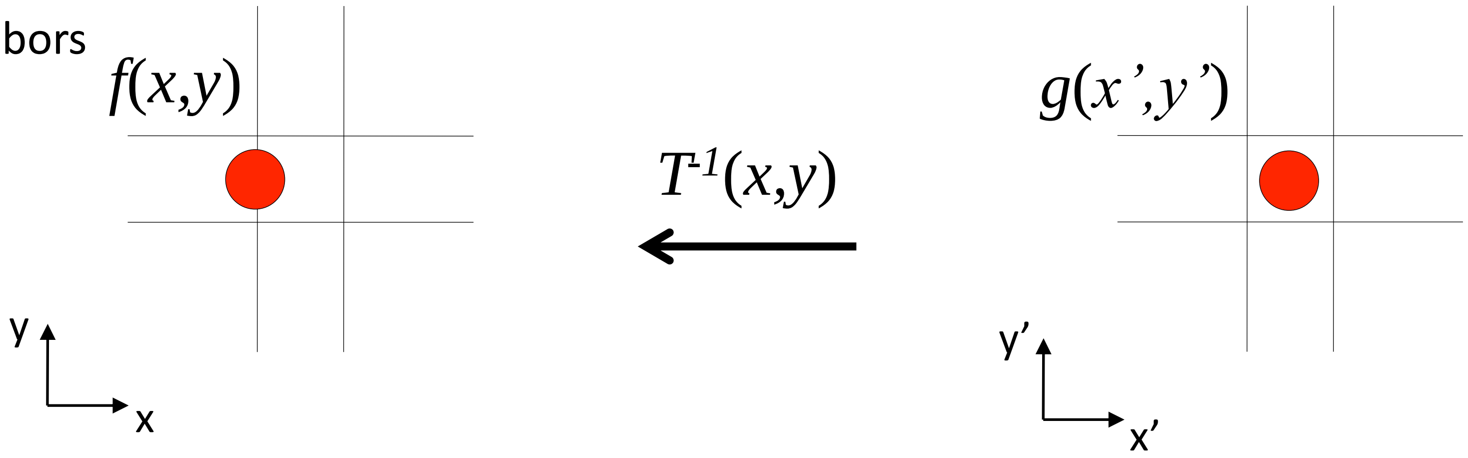
Inverse warping

Pixel may come from between two points

- How do we determine its intensity?

✓ Use interpolation

- Nearest Neighbors
- Bilinear
- Cubic
- Lanczos



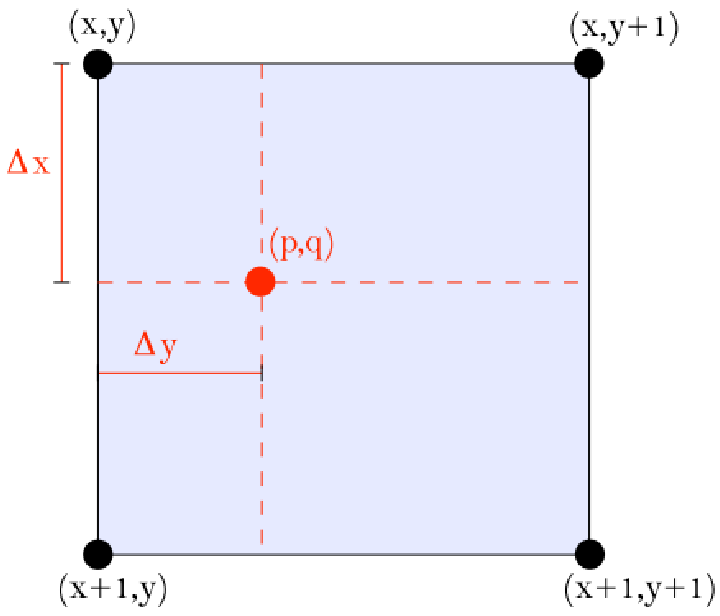
Nearest Neighbor Interpolation

The question

Let x and y be the integer coordinates of the lattice. What is the value of f at $\begin{bmatrix} p & q \end{bmatrix}^T$?

Nearest Neighbour Answer

$$\hat{f}(p,q) = f(\text{round}(p), \text{round}(q))$$



Problem with NN interpolation

Point Sampled: Aliasing!

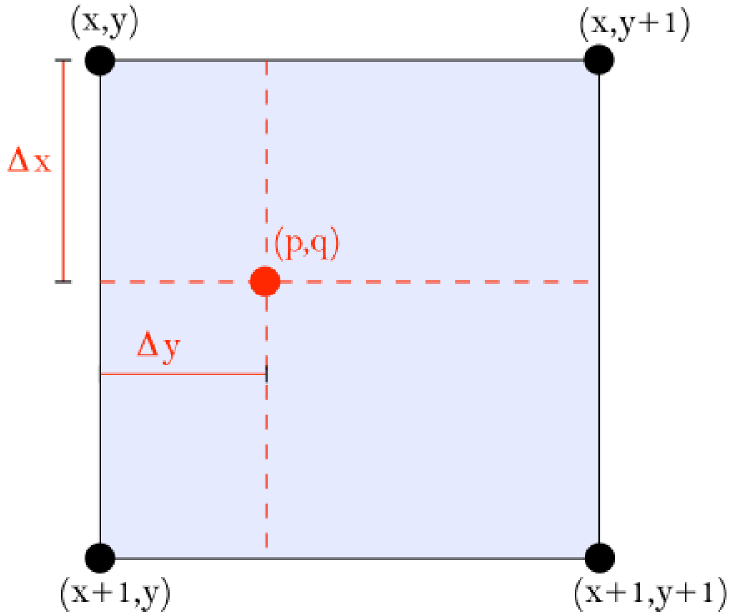
Correctly Bandlimited

Bilinear Interpolation

The question

Let x and y be the integer coordinates of the lattice. What is the value of f at $\begin{bmatrix} p & q \end{bmatrix}^T$?

- $F_{0,0} \stackrel{\text{def}}{=} f(x, y)$
- $F_{1,0} \stackrel{\text{def}}{=} f(x+1, y)$
- $F_{0,1} \stackrel{\text{def}}{=} f(x, y+1)$
- $F_{1,1} \stackrel{\text{def}}{=} f(x+1, y+1)$
- $\Delta x \stackrel{\text{def}}{=} p - x$ and $\Delta y \stackrel{\text{def}}{=} q - y$



Bilinear Interpolation

The question

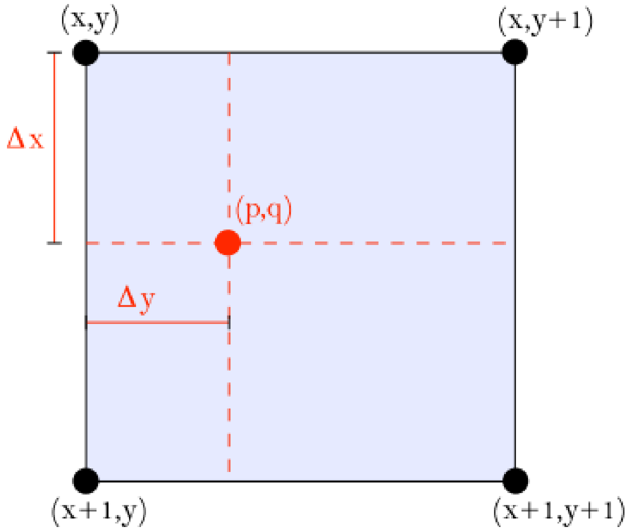
Let x and y be the integer coordinates of the lattice. What is the value of f at $\begin{bmatrix} p & q \end{bmatrix}^T$?

- Linear interpolation in the x direction:

$$\begin{aligned} f_y(\Delta x) &= (1 - \Delta x)F_{0,0} + \Delta x F_{1,0} \\ f_{y+1}(\Delta x) &= (1 - \Delta x)F_{0,1} + \Delta x F_{1,1} \end{aligned}$$

- Linear interpolation in the y direction:

$$\hat{f}(p, q) = (1 - \Delta y)f_y + \Delta y f_{y+1}$$



Bilinear Interpolation

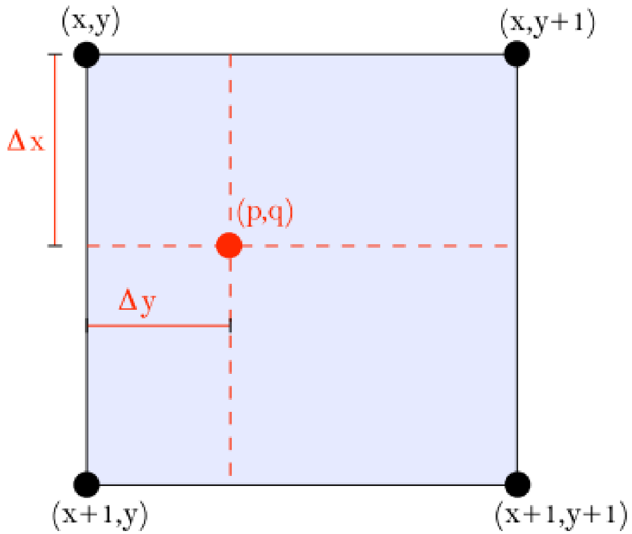
The question

Let x and y be the integer coordinates of the lattice. What is the value of f at $\begin{bmatrix} p & q \end{bmatrix}^T$?

Bilinear Interpolation Answer

Note that $\hat{f}(p,q)$ "passes through" the samples.

$$\begin{aligned}\hat{f}(p,q) = (1-\Delta y)(1-\Delta x)F_{0,0} &+ \\ (1-\Delta y)\Delta x F_{1,0} &+ \\ \Delta y(1-\Delta x)F_{0,1} &+ \\ \Delta y \Delta x F_{1,1}\end{aligned}$$



Forward vs inverse warping

Suppose we have two images.

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

y, x

$f(x, y)$

$T(x, y)$

$\longleftrightarrow$

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

y', x'

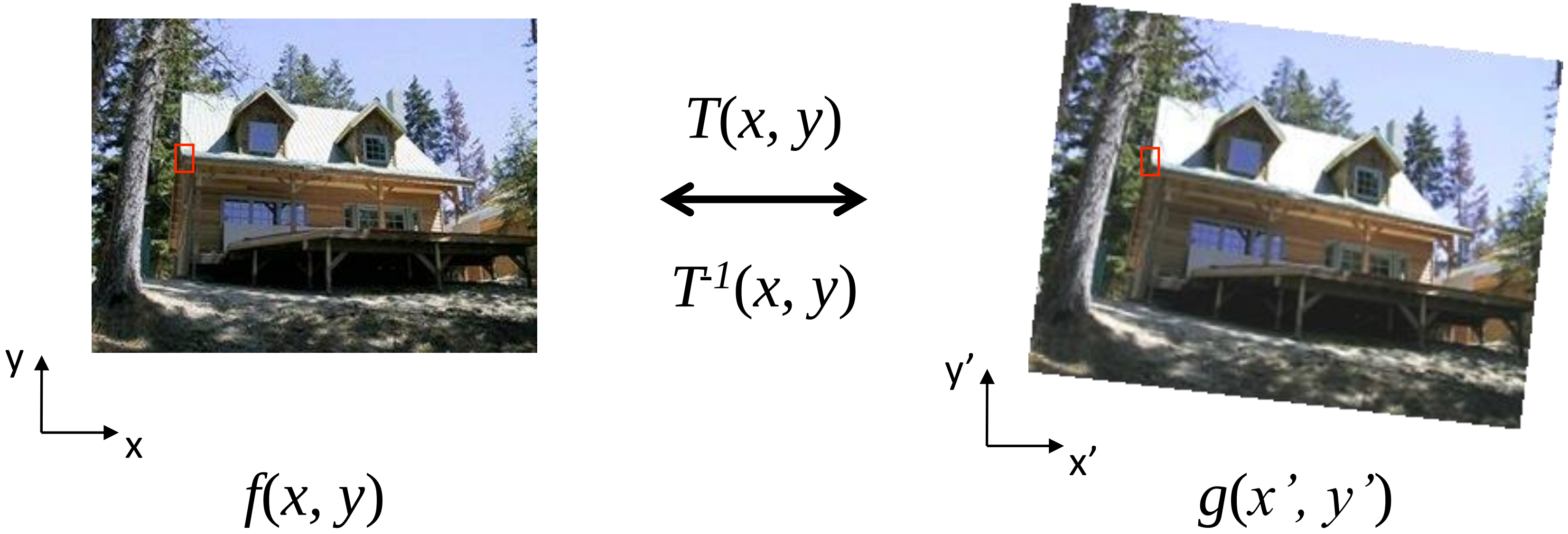
$g(x', y')$

Pros and cons of each?

Forward vs inverse warping

Suppose we have two images.

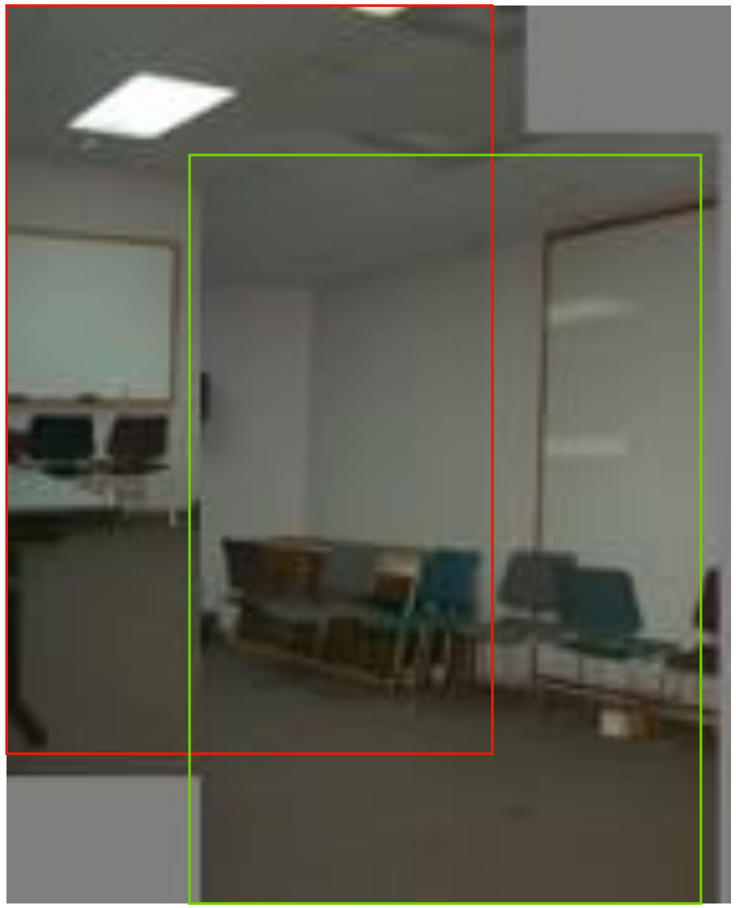
- How do we compute the transform that takes one to the other?



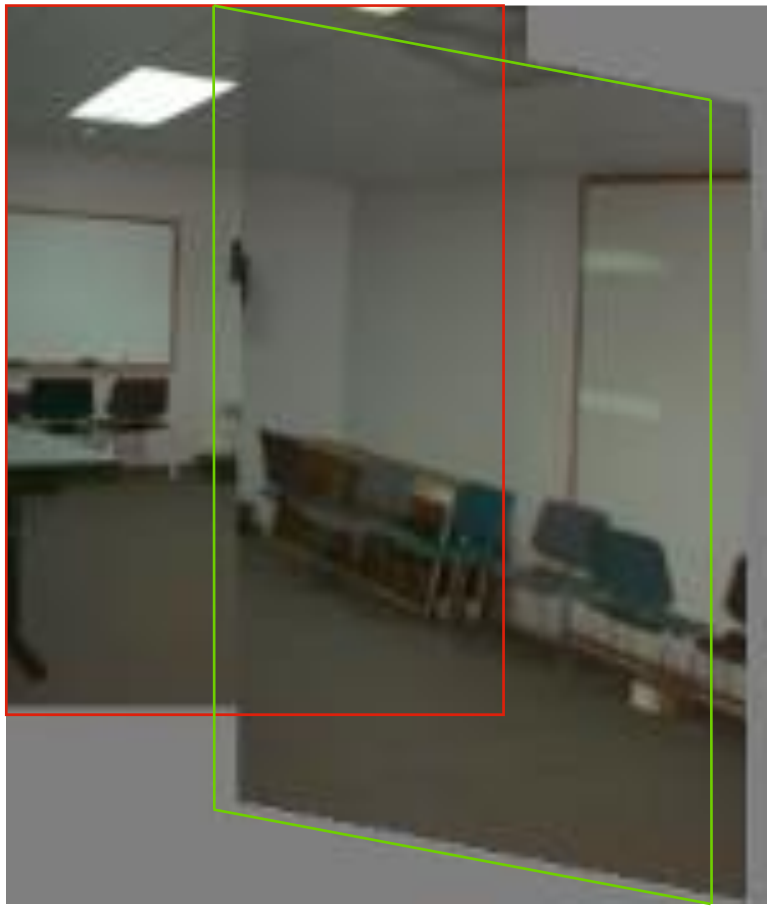
- Inverse warping eliminates holes in target image
- Forward warping does not require existence of inverse transform

Warping with different transformations

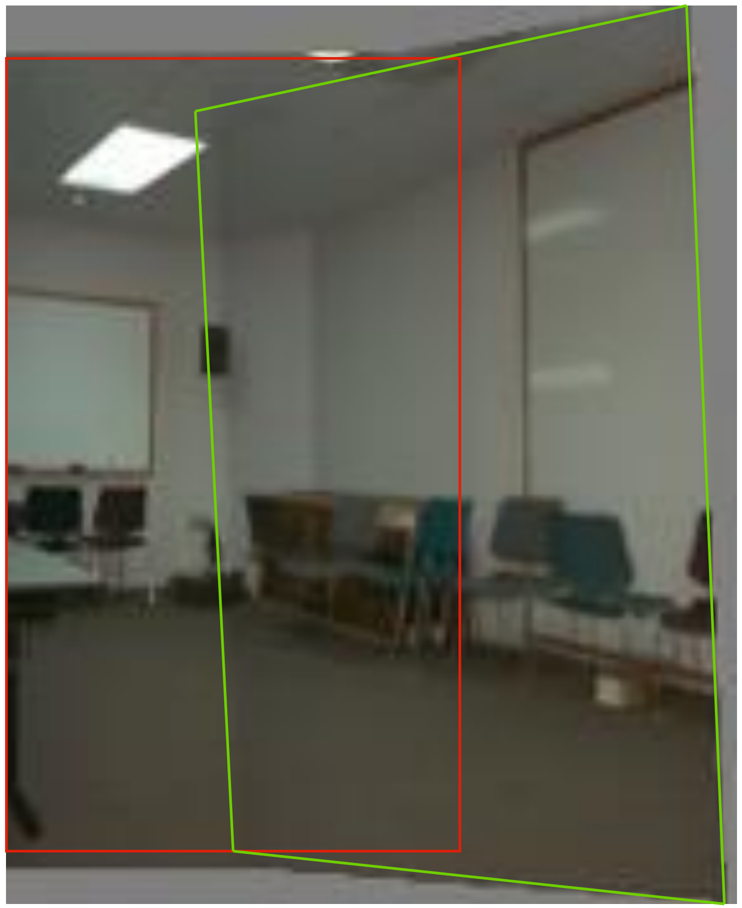
translation



affine



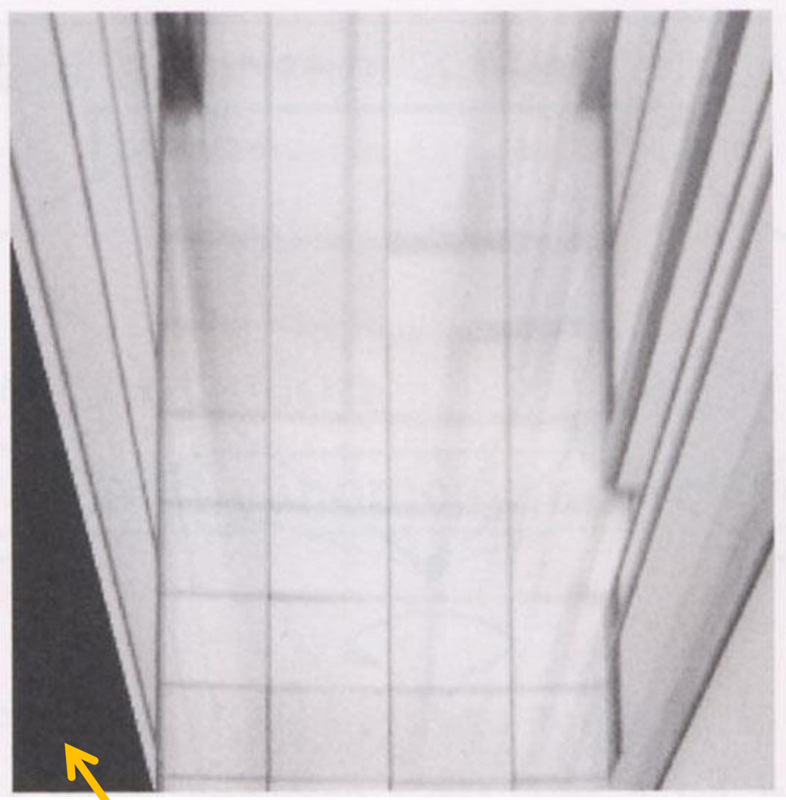
pProjective (homography)



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

Understanding geometric patterns

What is the pattern on the floor?

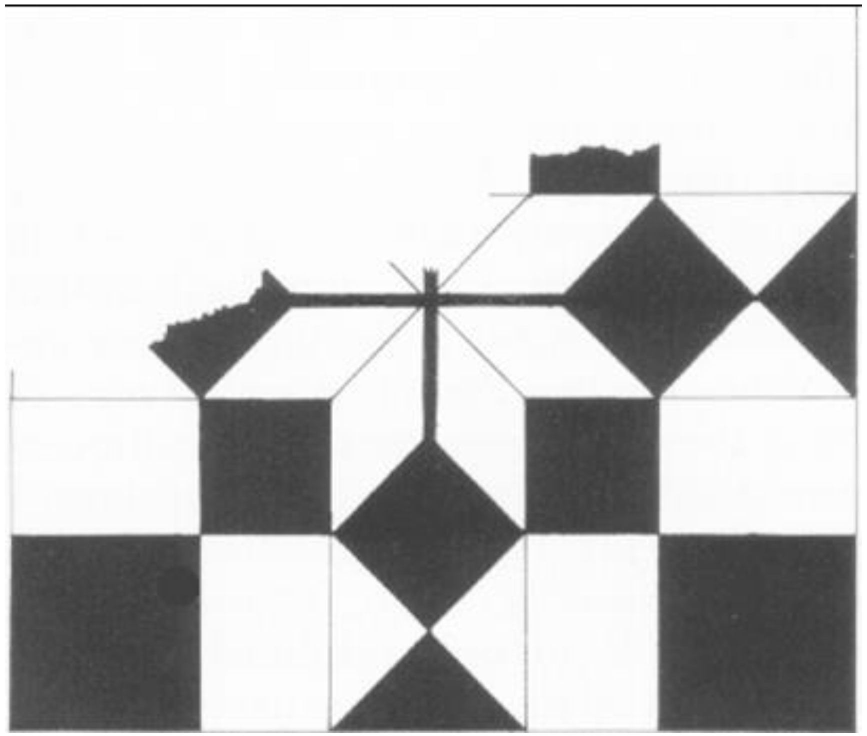
magnified view of floor

Understanding geometric patterns

What is the pattern on the floor?

magnified view of floor

rectified view

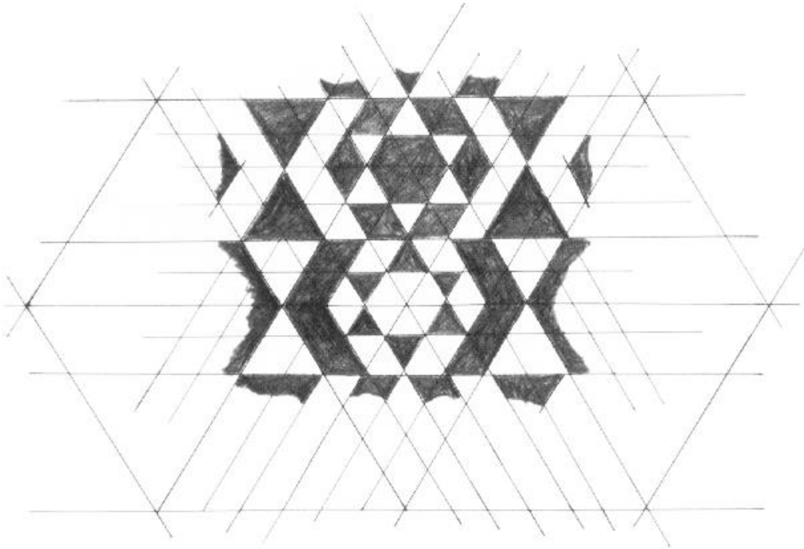


reconstruction from rectified view

Understanding geometric patterns

Very popular in renaissance drawings (when perspective was discovered)

rectified view of floor



reconstruction

A weird drawing

Holbein, “The Ambassadors”

A weird drawing

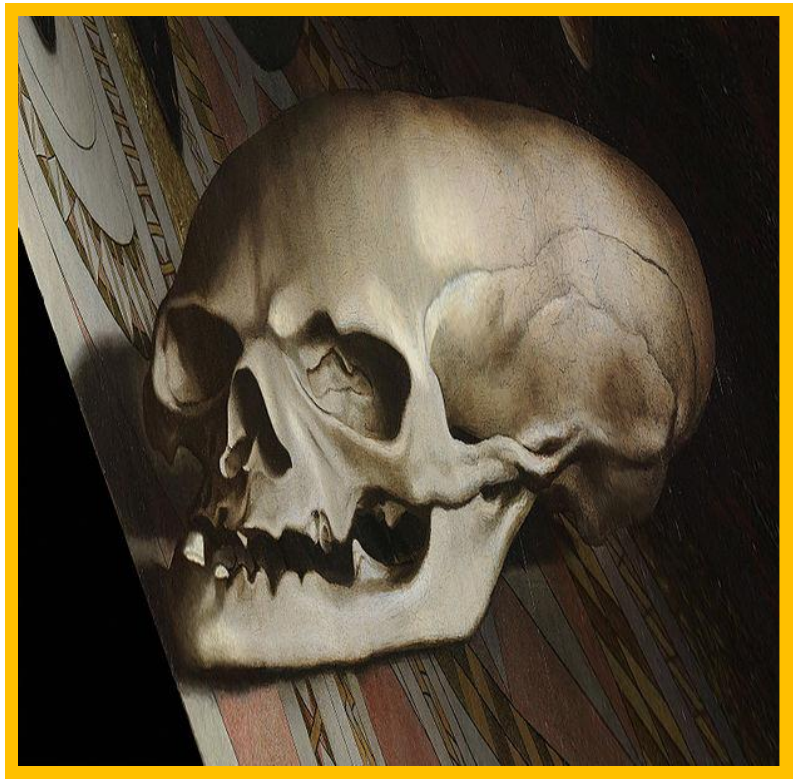
Holbein, “The Ambassadors”

What’s this???

A weird drawing

Holbein, “The Ambassadors”

skull under anamorphic perspective



rectified view

A weird drawing

Holbein, “The Ambassadors”

DIY: use a polished spoon to see the skull

Panoramas from image stitching

1. Capture multiple images from different viewpoints.

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

References

Basic reading:

- Szeliski textbook, Section 3.6.

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, and also a very useful reference for the second part of the class.
- 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).

Questions?