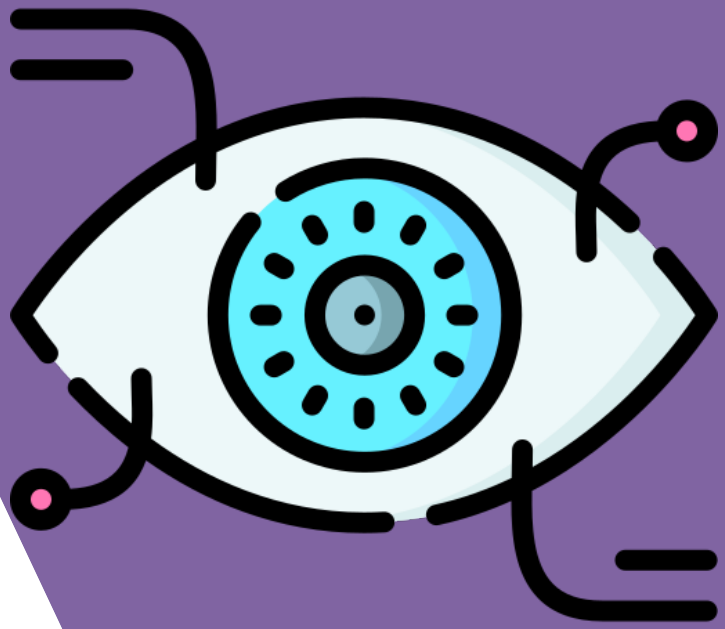
Computer Vision

A.A. 2024-20245

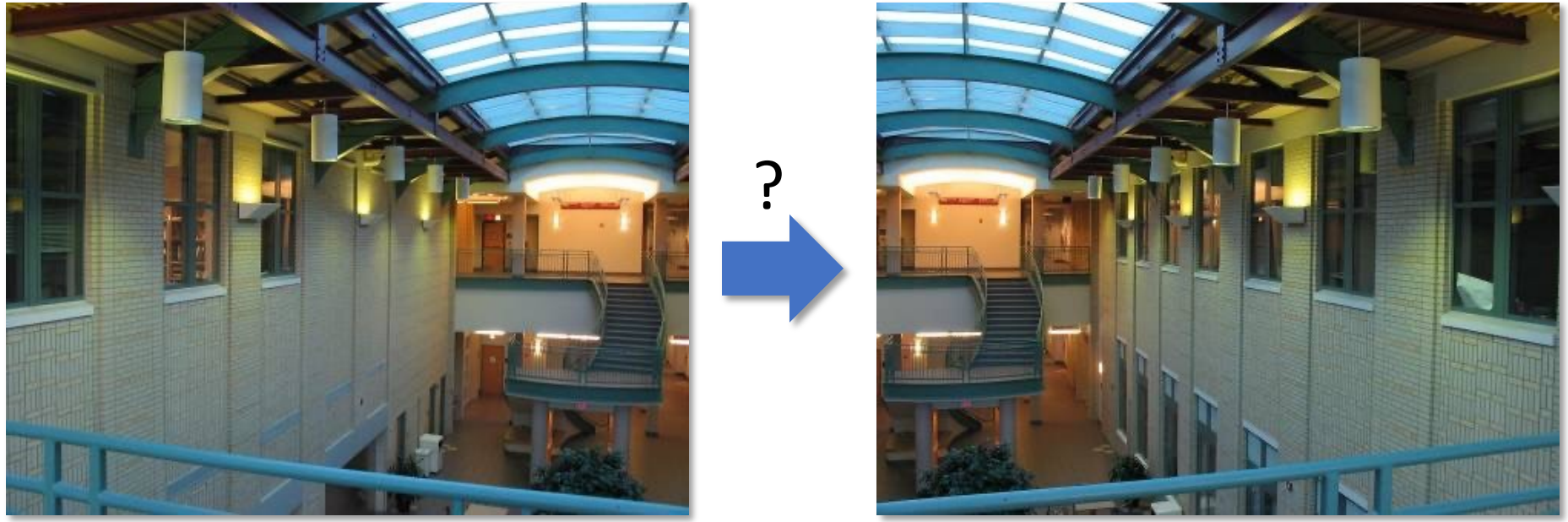
Lecture 7: 2D Transformation and Alignment



SAPIENZA
UNIVERSITÀ DI ROMA

ALCOR Lab

What is the geometric relationship between these two images?



What is the geometric relationship between these two images?

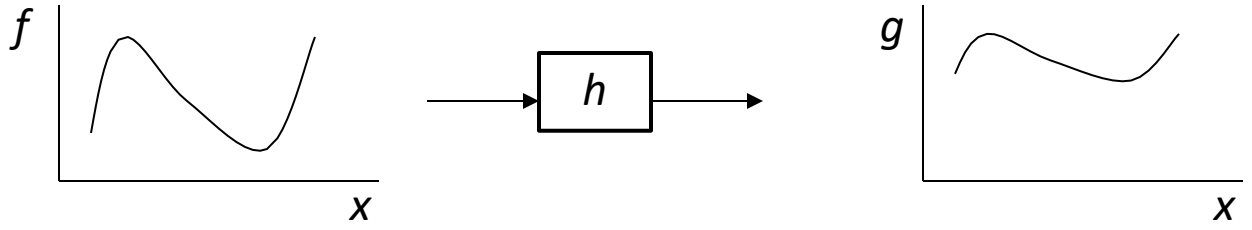
Very important for creating mosaics!

First, we need to know what this transformation is.

Second, we need to figure out how to compute it using feature matches.

Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$

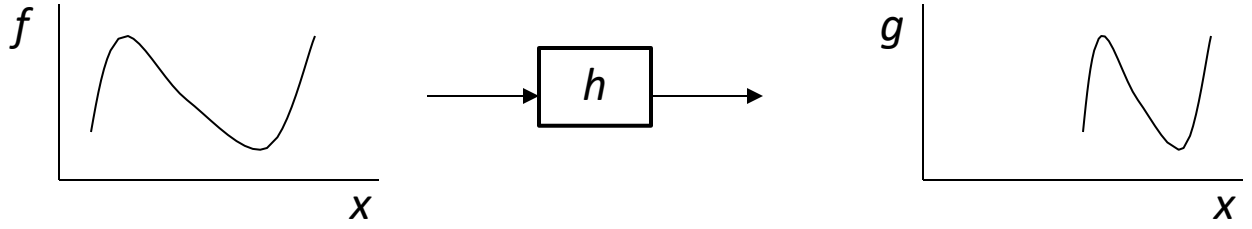
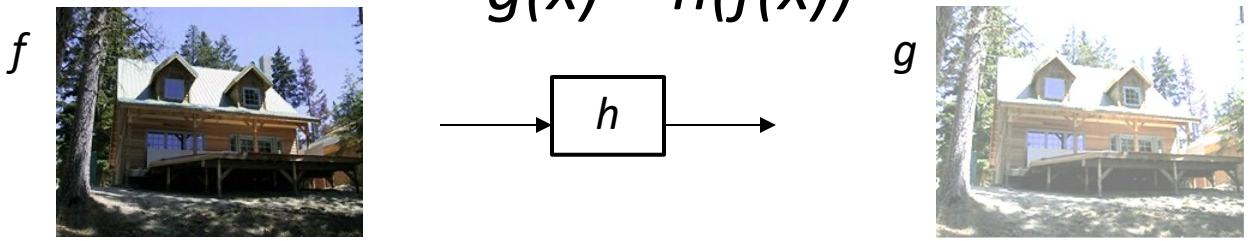
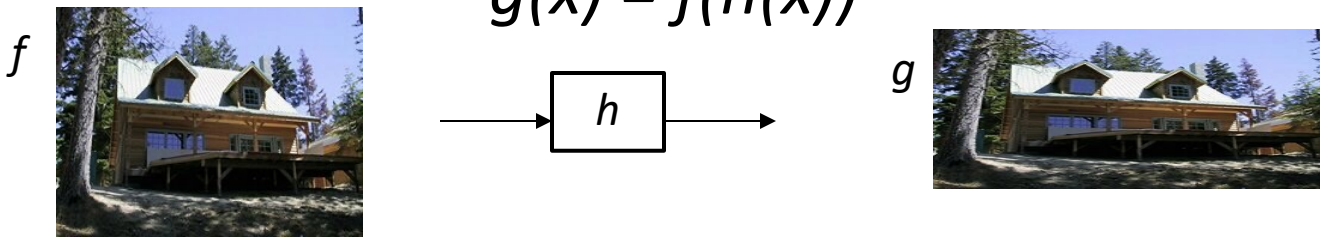


Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$



Parametric (global) warping

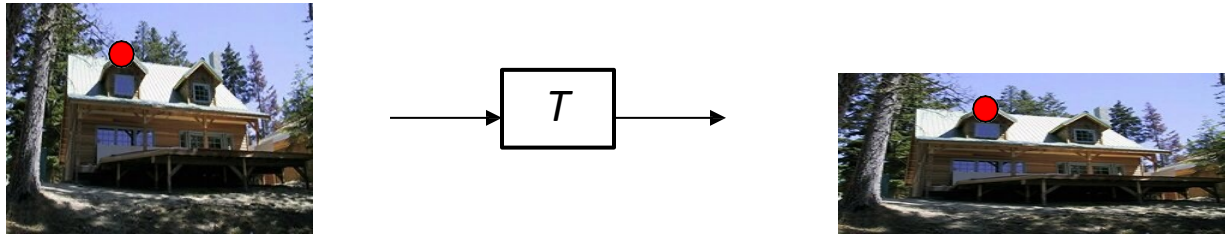
- Examples of parametric warps:

translation

rotation

aspect

Parametric (global) warping



p = (x,y)

p' = (x',y')

- Transformation T is a coordinate-changing machine:

$$\mathbf{p}' = T(\mathbf{p})$$

- What does it mean that *T* is global?
  - Is the same for any point **p**
  - can be described by just a few numbers (parameters)
- Let's consider *linear* transforms (can be represented by a 2x2 matrix):

$$\mathbf{p}' = \mathbf{T}\mathbf{p} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Today’s class

- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

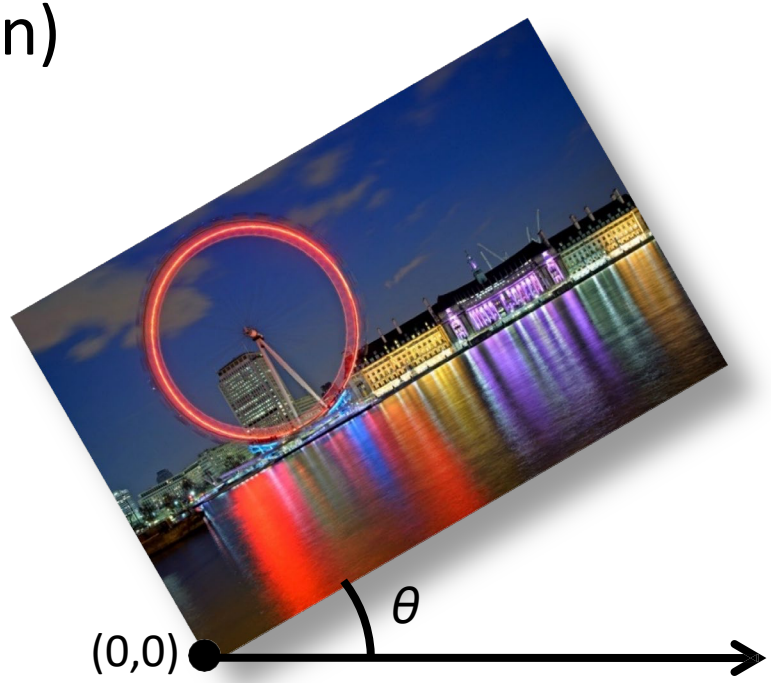
- Computing 2D Transformations
  - Linear Least Squares
  - Affine
  - Perspective (Homography)

Today’s class

- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)
- Computing 2D Transformations
  - Linear Least Squares
  - Affine
  - Perspective (Homography)

Common linear transformations

- Rotation by angle θ (about the origin)



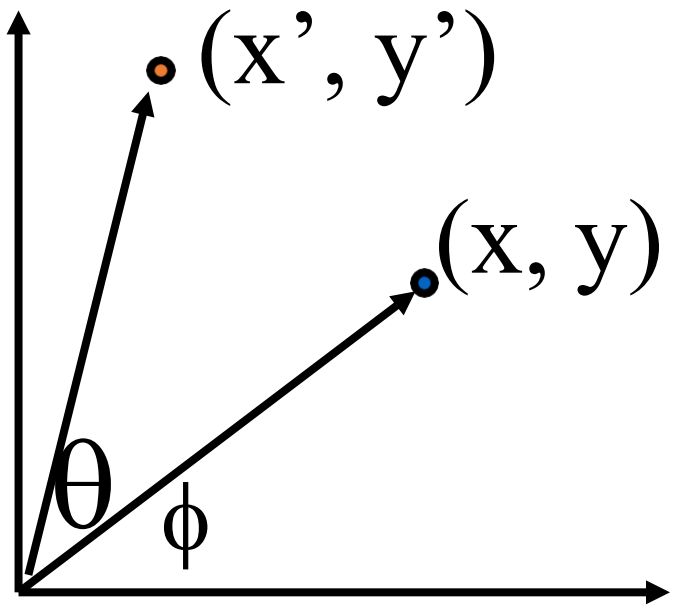
$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

What is the inverse?

For rotations:

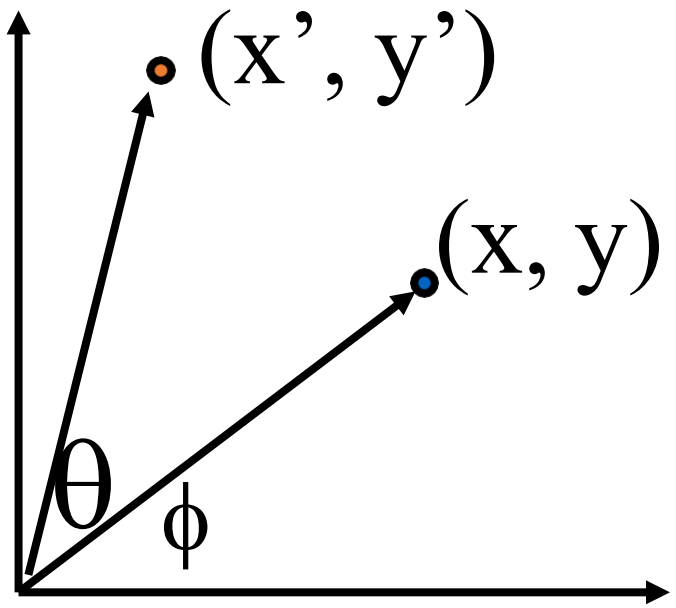
$$\mathbf{R}^{-1} = \mathbf{R}^{T}$$

2-D Rotation



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

2-D Rotation



$x = r \cos(\phi)$

$y = r \sin(\phi)$

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

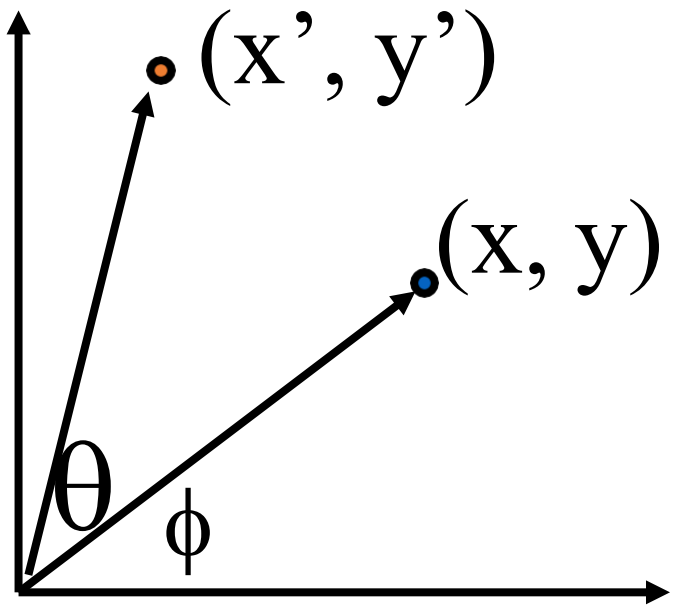
$y' = r \sin(\phi + \theta)$

Trig Identity…

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

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

2-D Rotation



$x = r \cos(\phi)$

$y = r \sin(\phi)$

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

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

Trig Identity…

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

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

Substitute…

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

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

2x2 Matrices

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

2D Identity?

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

2D Scale around (0,0)?

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

Common linear transformations

- Uniform scaling by *s*:

$$\mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

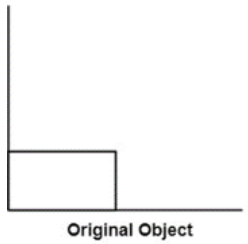
2x2 Matrices

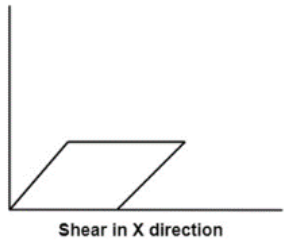
- What types of transformations can be represented with a 2x2 matrix?

2D Rotate around (0,0)?

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

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

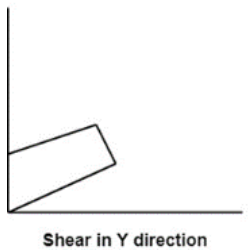


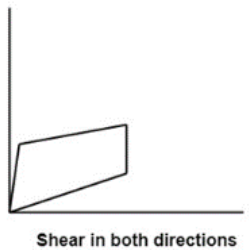


2D Shear?

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

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





2x2 Matrices

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

2D Mirror about Y axis?

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

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

2D Mirror over (0,0)?

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

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

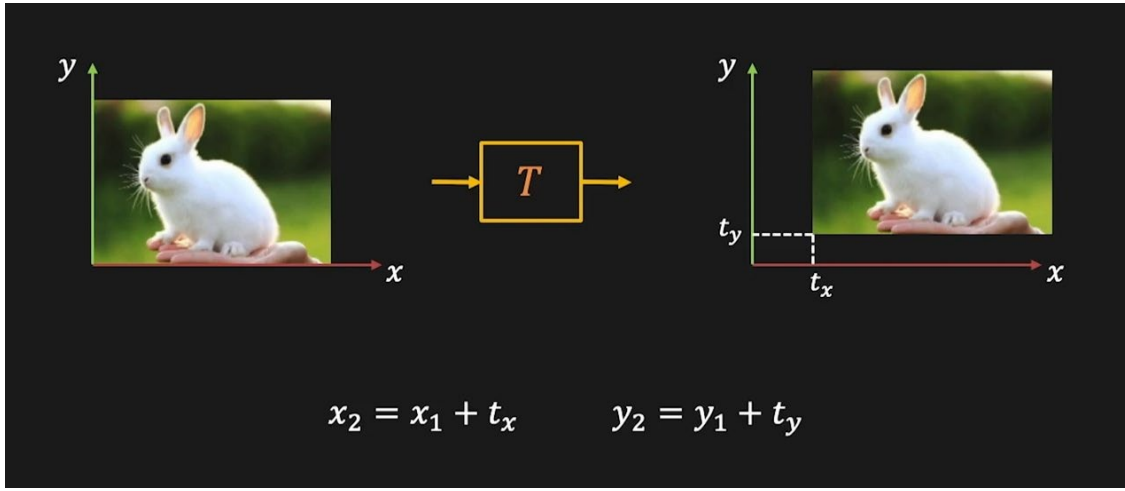
2x2 Matrices

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

2D Translation?

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

NO!



Translation is not a linear operation on 2D coordinates

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
  - Ratios are preserved
  - Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix}}_{T'=T1T2T3} \begin{bmatrix} x \\ y \end{bmatrix}$$

Today’s class

- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)
- Computing 2D Transformations
  - Linear Least Squares
  - Affine
  - Perspective (Homography)

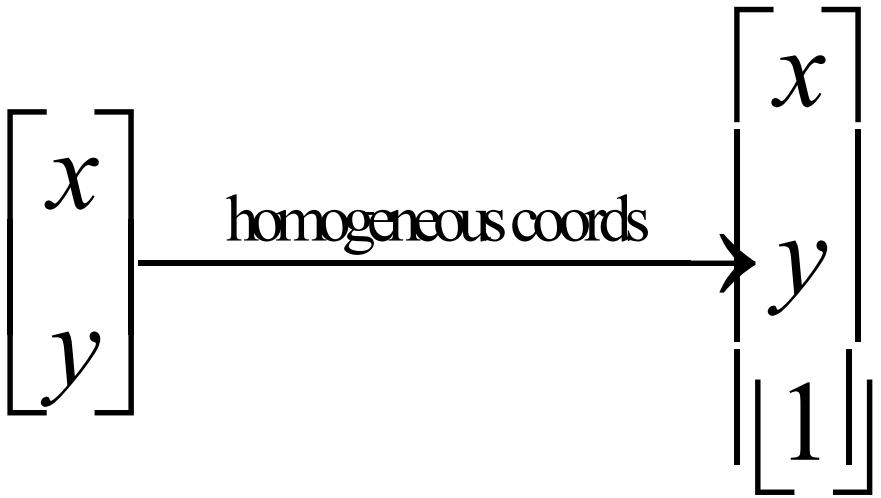
Homogeneous Coordinates

- **Q: How can we represent translation as a 3x3 matrix?**

$$x' = x + t_x$$

$$y' = y + t_y$$

- ***Homogeneous coordinates***
  - represent coordinates in 2 dimensions with a 3-vector

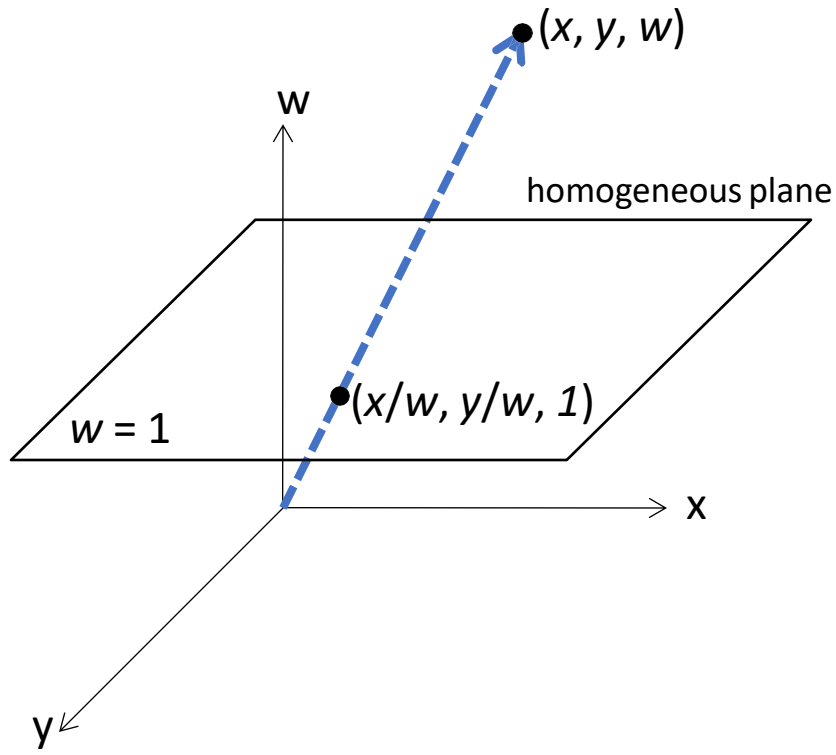


Homogeneous coordinates

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

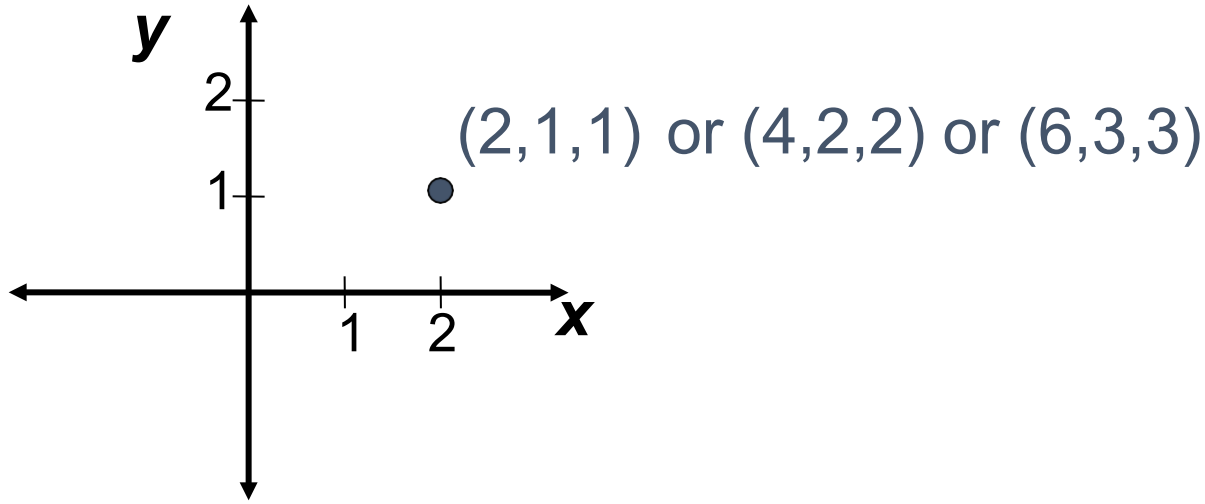


Converting *from* homogeneous coordinates

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

Homogeneous Coordinates

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



Convenient coordinate system to represent many useful transformations

Homogeneous Coordinates

- **Q: How can we represent translation as a 3x3 matrix?**

$$x' = x + t_x$$

$$y' = y + t_y$$

- A: Using the rightmost column:

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

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

Affine transformations

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

any transformation represented by a 3x3 matrix with last row [0 0 1] we call an *affine* transformation

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

Basic affine transformations

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

Translate

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

Scale

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

2D *in-plane* rotation

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

Shear

Matrix Composition

- Transformations can be combined by matrix multiplication

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

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

Affine transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

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

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition: if you take any two transformations from a set and compose them (apply them one after the other), the result is still a transformation from that same set

Today’s class

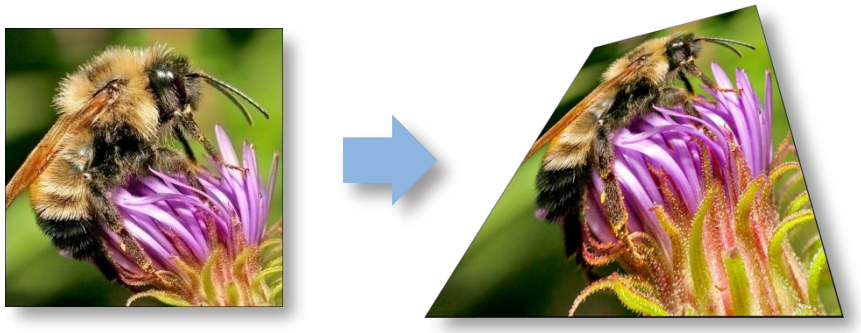
- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

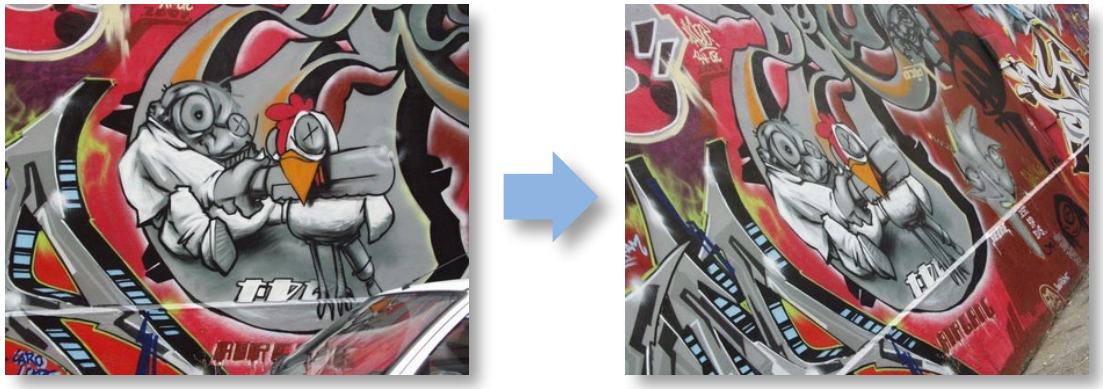
- Computing 2D Transformations
  - Linear Least Squares
  - Affine
  - Perspective (Homography)

Projective Transformations *aka* Homographies *aka* Planar Perspective Maps

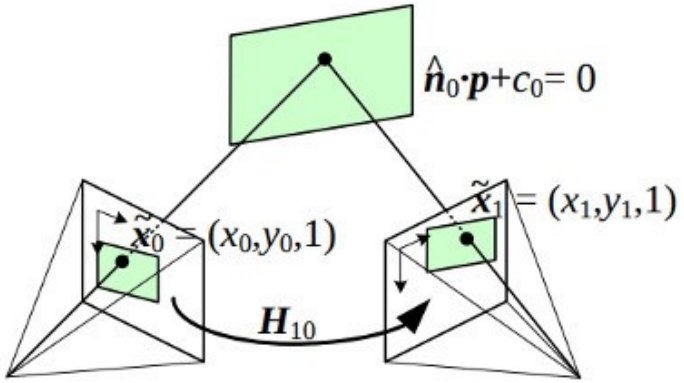
$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

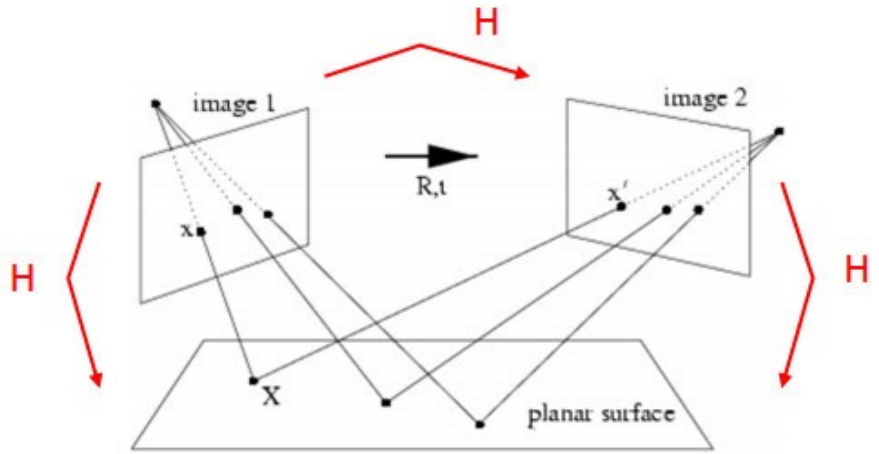
Called a *homography*
(or *planar perspective map*)





Projective Transformations *aka* Homographies *aka* Planar Perspective Maps



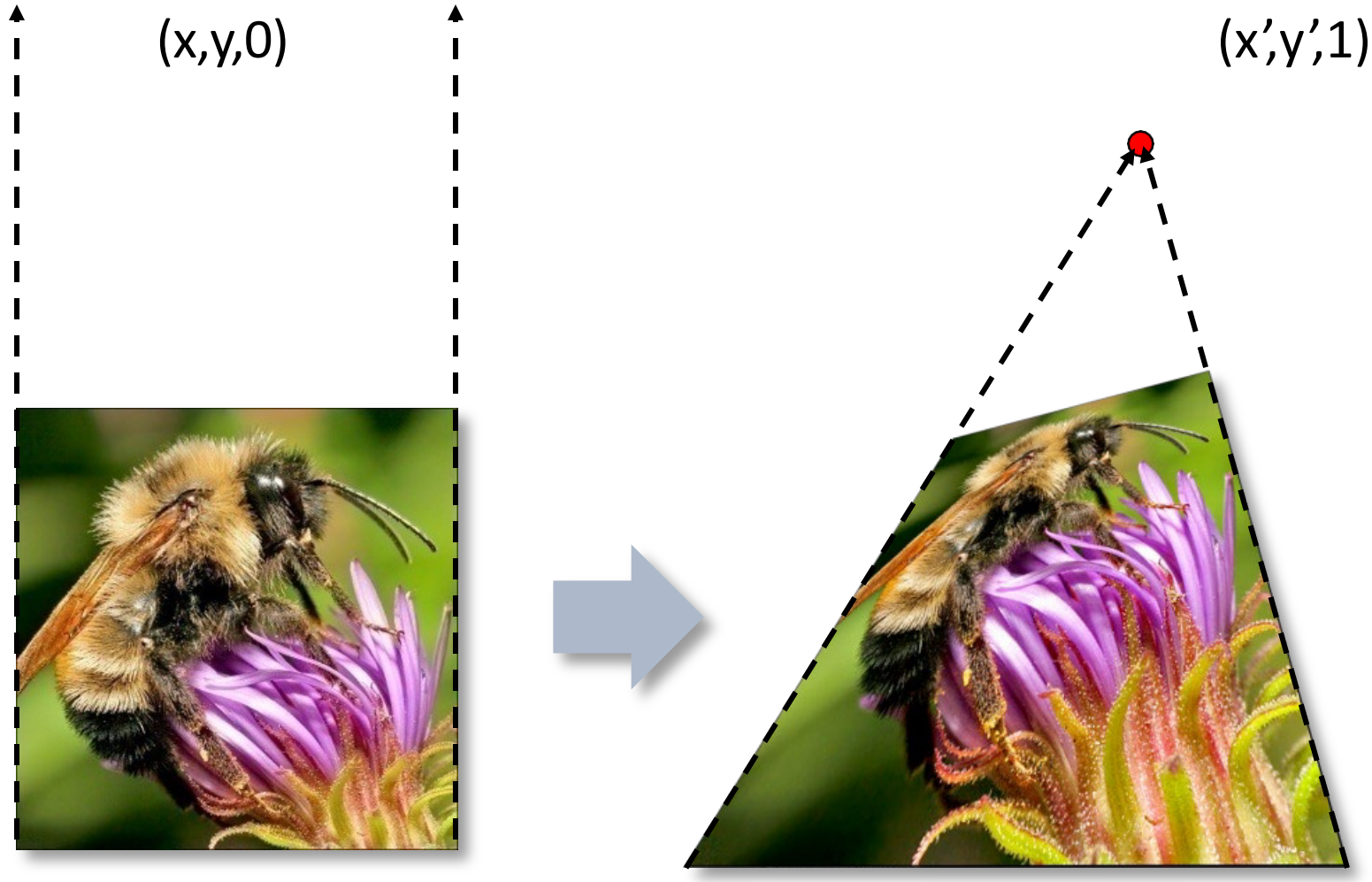


$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Any two images of the same planar surface in 3D space are related by a homography (assuming a pinhole camera model).

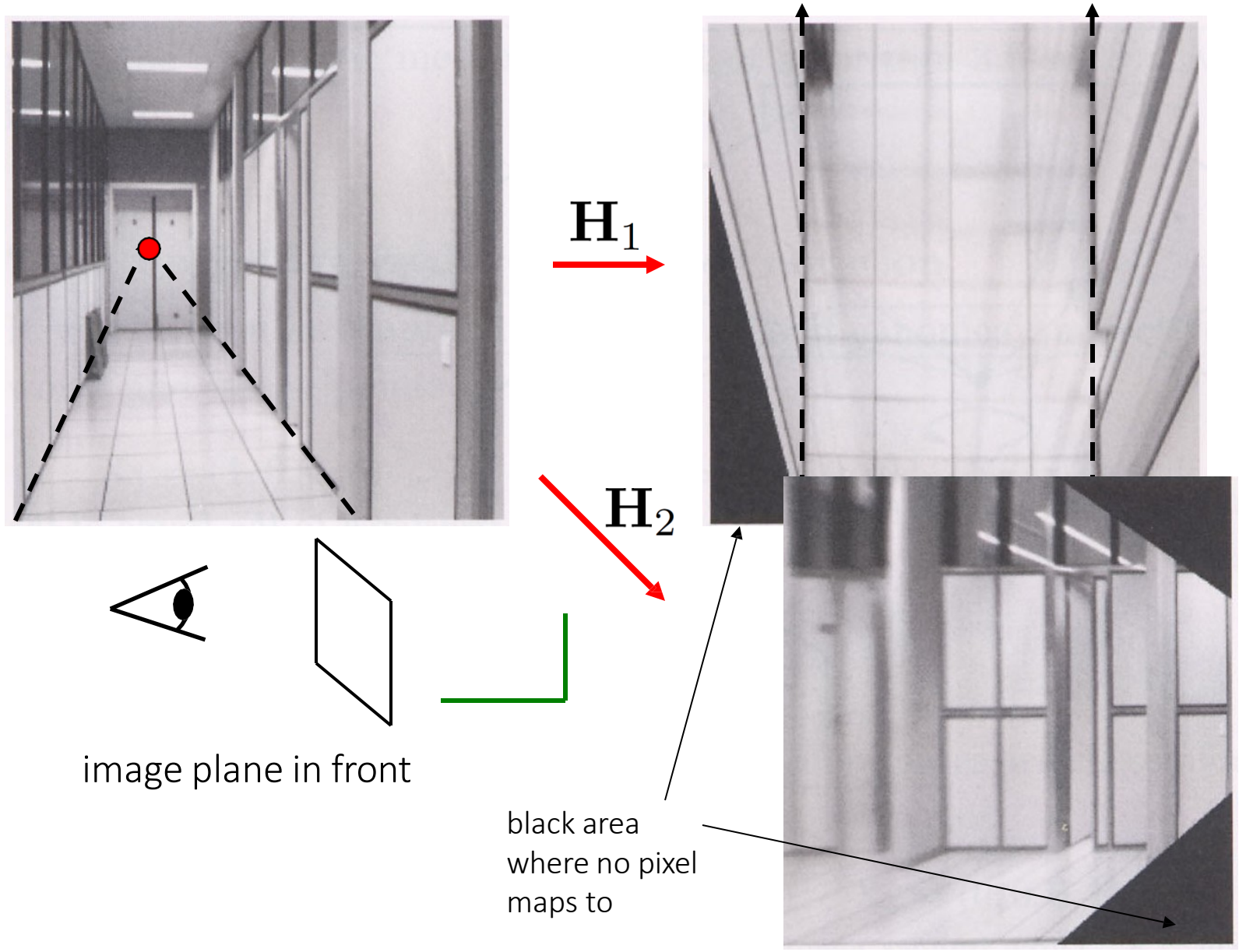
A homography is a transformation that maps points from one plane to another.

Points at infinity



In the context of homographies are used to represent vanishing points or parallel lines that converge at infinity in the original scene in homogeneous coordinates.

Image warping with homographies



Homographies (Projective Transformation)

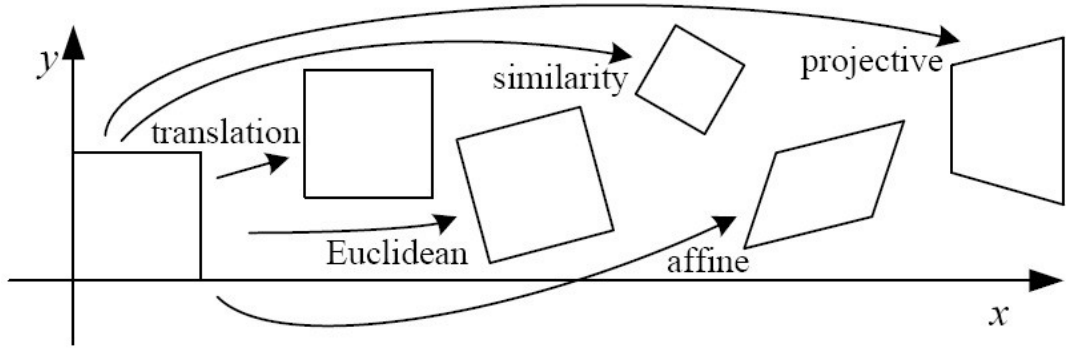
- Homographies …
  - Affine transformations, and
  - Projective warps

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

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition

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	

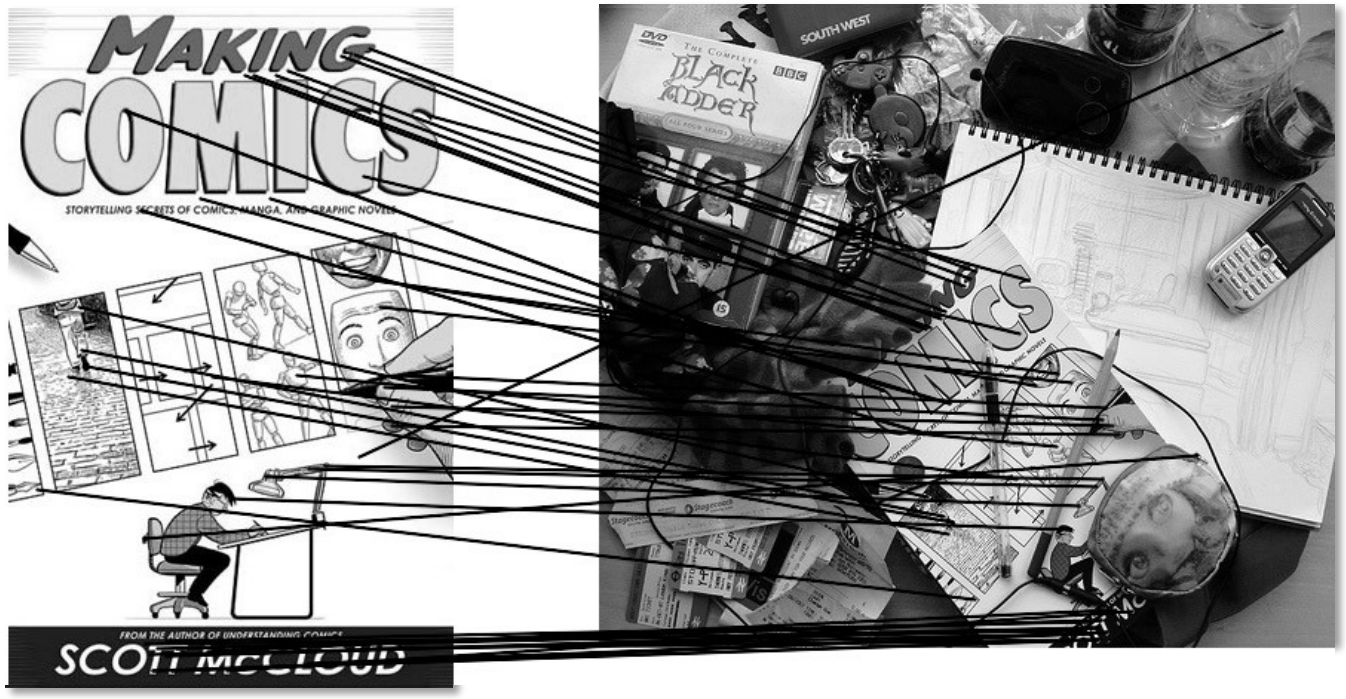
Today’s class

- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

- Computing 2D Transformations
  - Linear Least Squares
  - Affine
  - Perspective (Homography)

Computing transformations

- Given a set of matches between images A and B
  - How can we compute the transform T from A to B?



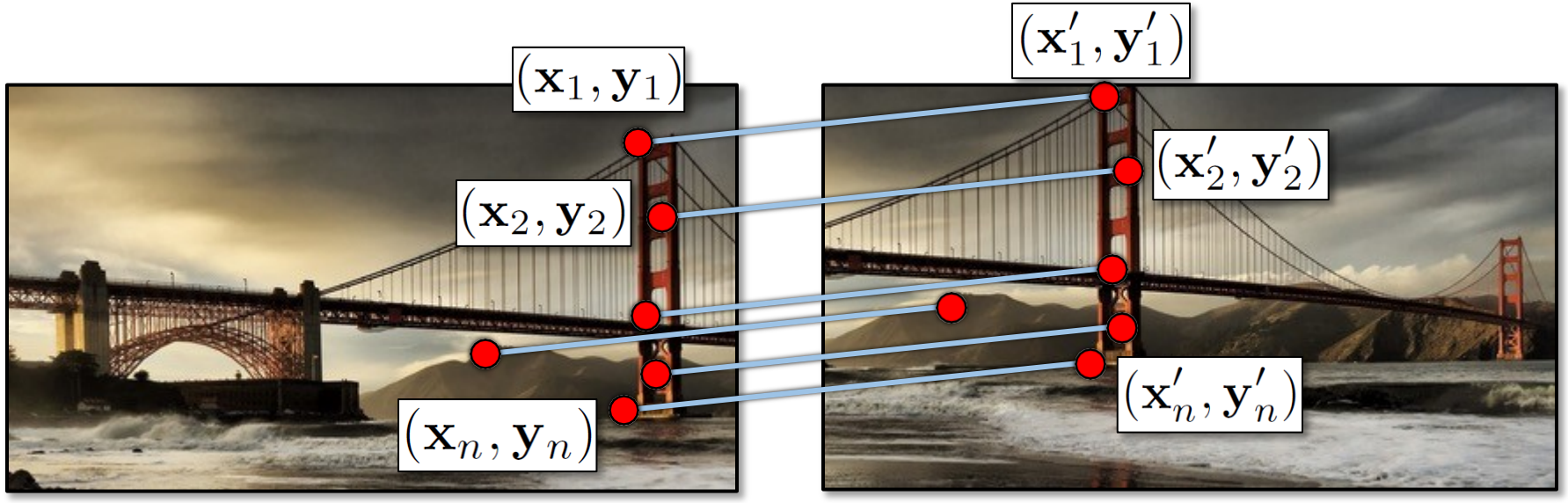
  - Find transform T that best “agrees” with the matches

Today’s class

- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

- Computing 2D Transformations
  - Linear Least Squares
  - Affine
  - Perspective (Homography)

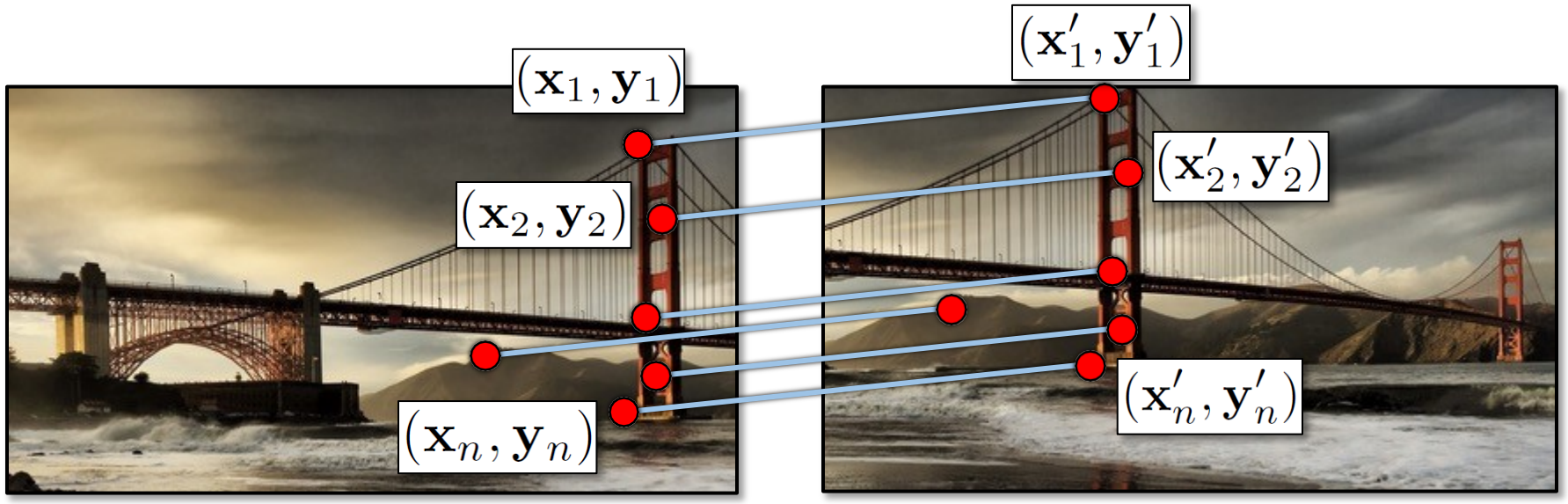
Simple case: translations



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

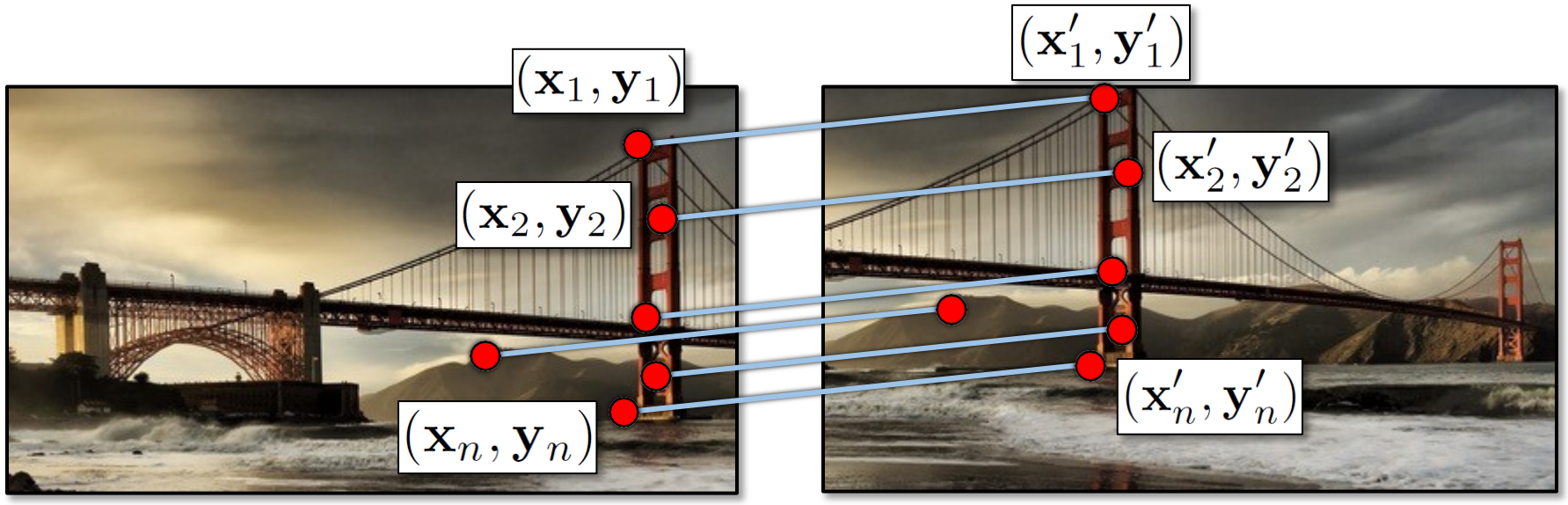


$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

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

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many matches do we need?

Another view

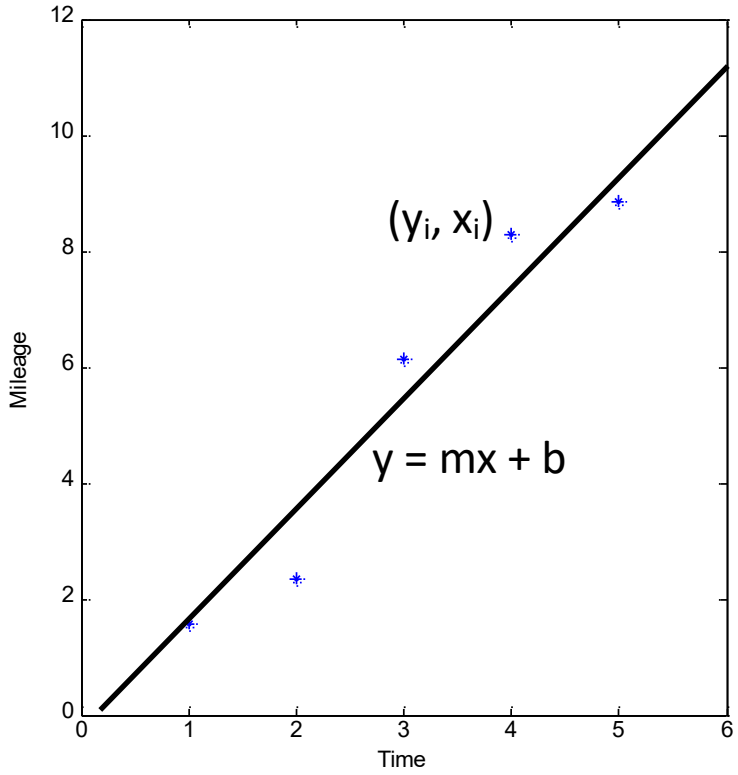


$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

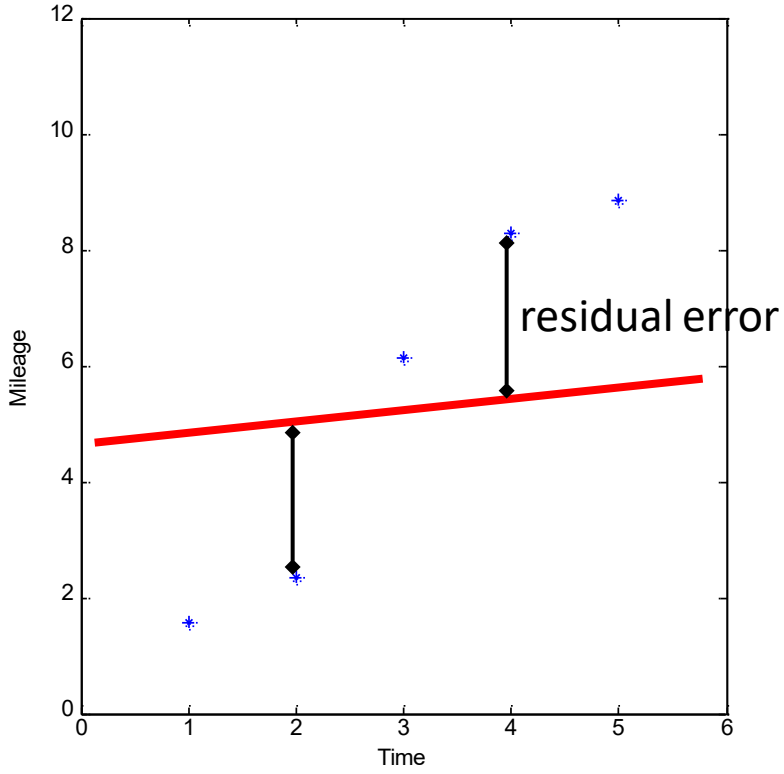
$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- Problem: more equations than unknowns
  - "Overdetermined" system of equations: two matching points are sufficient to compute the translation transformation.
  - If you have more than two points, you can use the method of least squares to obtain a more robust estimation of the translation parameters.

Least squares: linear regression



Linear regression



$$\text{Cost}(m, b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2$$

Calculate partial derivatives w.r.t. m and b and set them to 0.

Linear regression

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Least squares

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

- Find **t** that minimizes

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

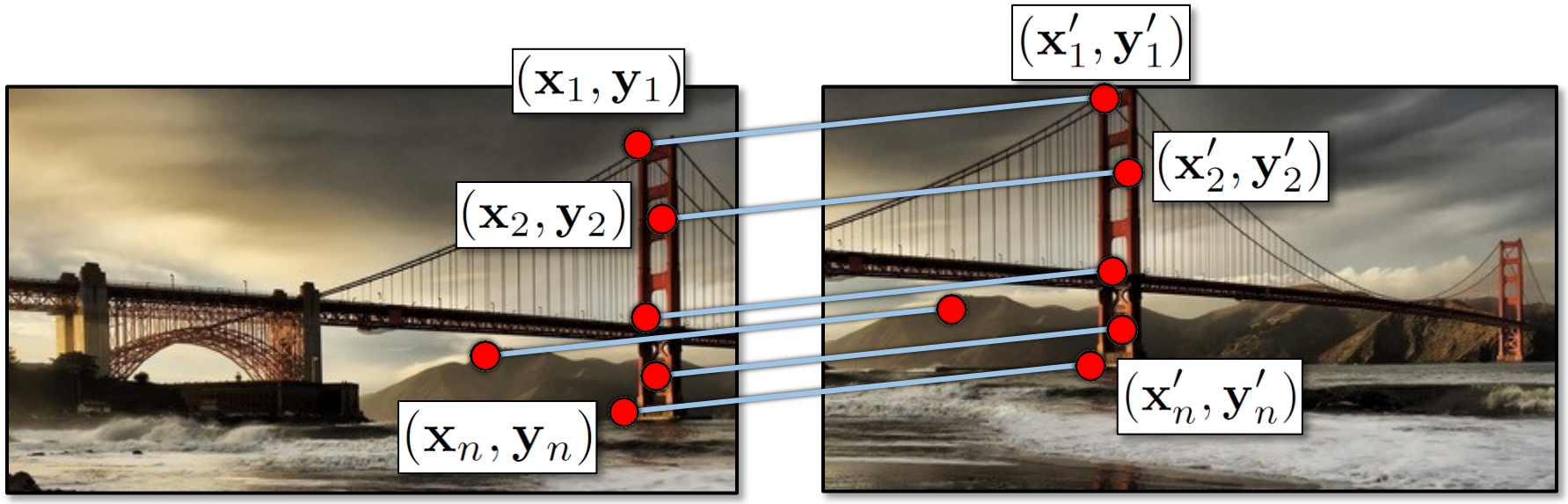
- To solve, form the *normal equations*

NB: Unique least square solution, column of A linearly indipendent, Atranspose and A are invertible

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

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

Another view



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

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

- 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

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

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

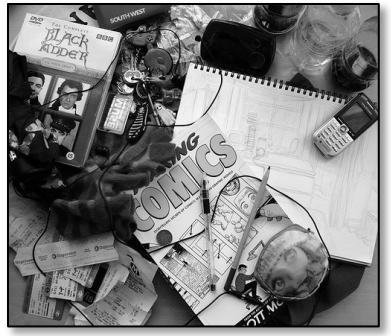
Today’s class

- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

- Computing 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

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:

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

- Cost function:

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

Calculate partial derivatives w.r.t. (a,b,c,d,e,f) and set to 0.

Affine transformations

- Matrix form

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

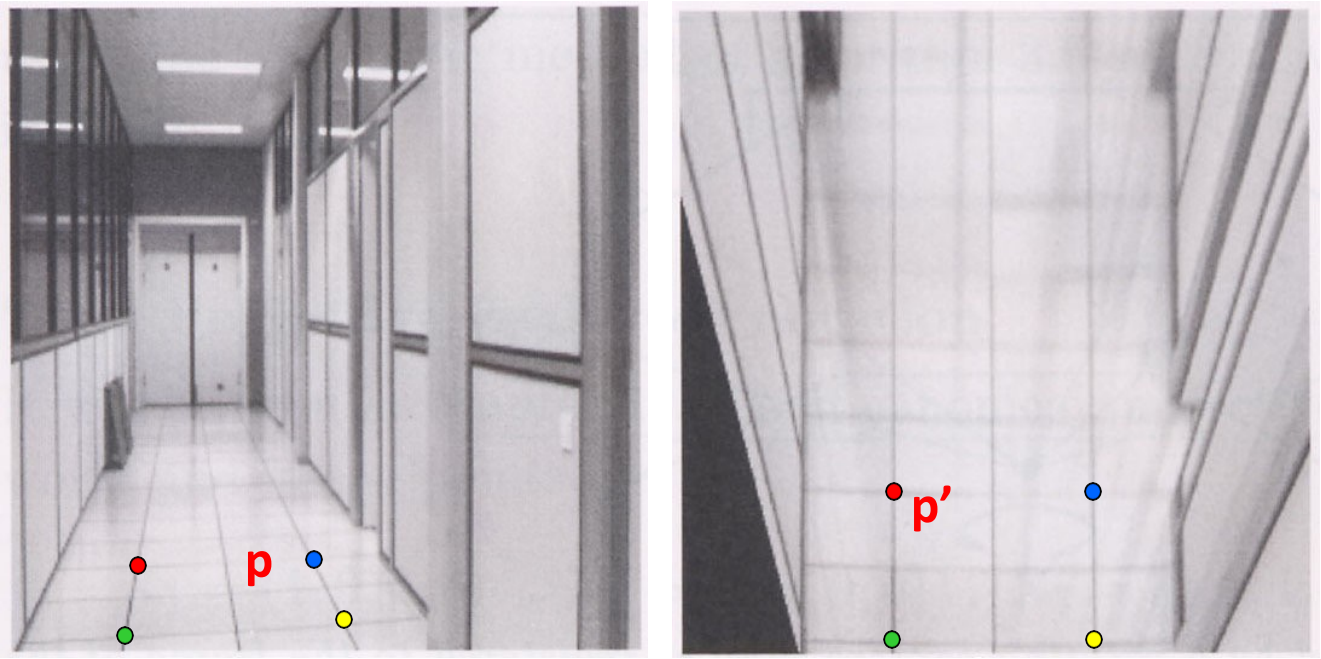
$$\underset{2n \times 6}{\mathbf{A}} \quad \underset{6 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

Today’s class

- Types of 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

- Computing 2D Transformations
  - Linear
  - Affine
  - Perspective (Homography)

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 matches are necessary to solve for **H**?

Solving for homographies

For a given pair of corresponding points

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

Not linear!

Rearranging the terms

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$
$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

Solving for homographies

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

Rearranging the terms and writing as linear equation

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

Known

Unknown

Solving for homographies

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$\mathbf{A}$ (2n × 9) $\mathbf{h}$ (9) $\mathbf{0}$ (2n)

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

- Since *h* is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$ ||h||^2=1
- **Constrained Least Square Formulation**

Solving for homographies

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

Rank(A) = 8

- Calculate Singular Value decomposition of A -> $A = UDV^T$
- U is 2n x r ; D is r x r (diagonal matrix with singular values) ; V is 9 x r, where r=rank(A).
- In ideal case r=rank(A)=8, h is in null-space of A.
- In practice rank(A)=9, thus the goal is to find the smallest singular value of A.
- Smallest singular value of A also indicates how well the homography can be estimated.
- A singular value is the positive square root of an eigenvalue of A^TA
- Calculate $A^TA = VDU^TUDV^T = V D^2 V^{-1}$ (Since, $U^TU = V^TV = I$)
- This is eigen-decomposition of A^TA
- Smallest singular value of A -> Smallest eigen value of A^TA.
- Solution: **optimal *h* = eigenvector of A^TA with smallest eigenvalue**.

Computing transformations

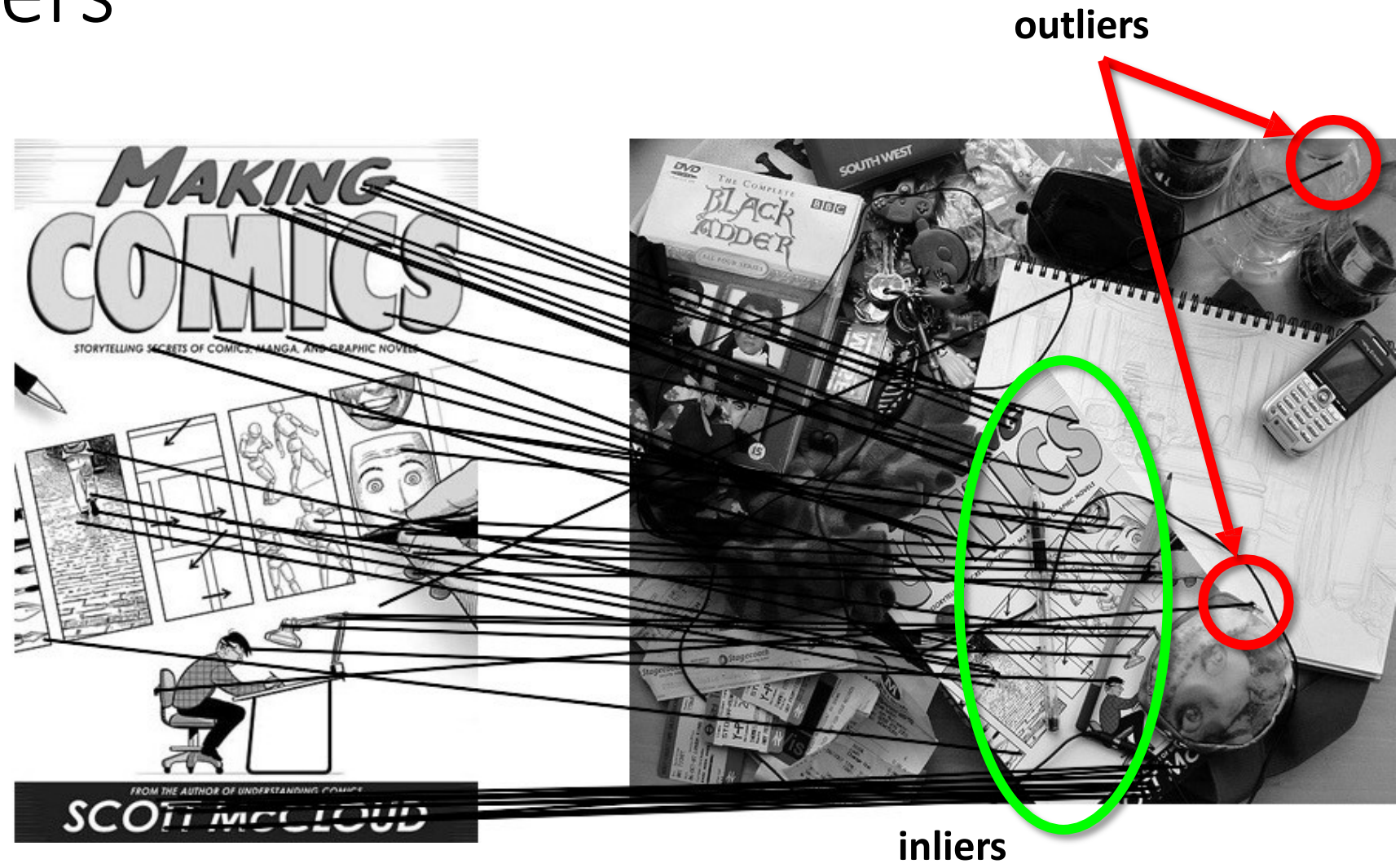
Image alignment algorithm

Given images A and B

1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

What could go wrong?

Outliers

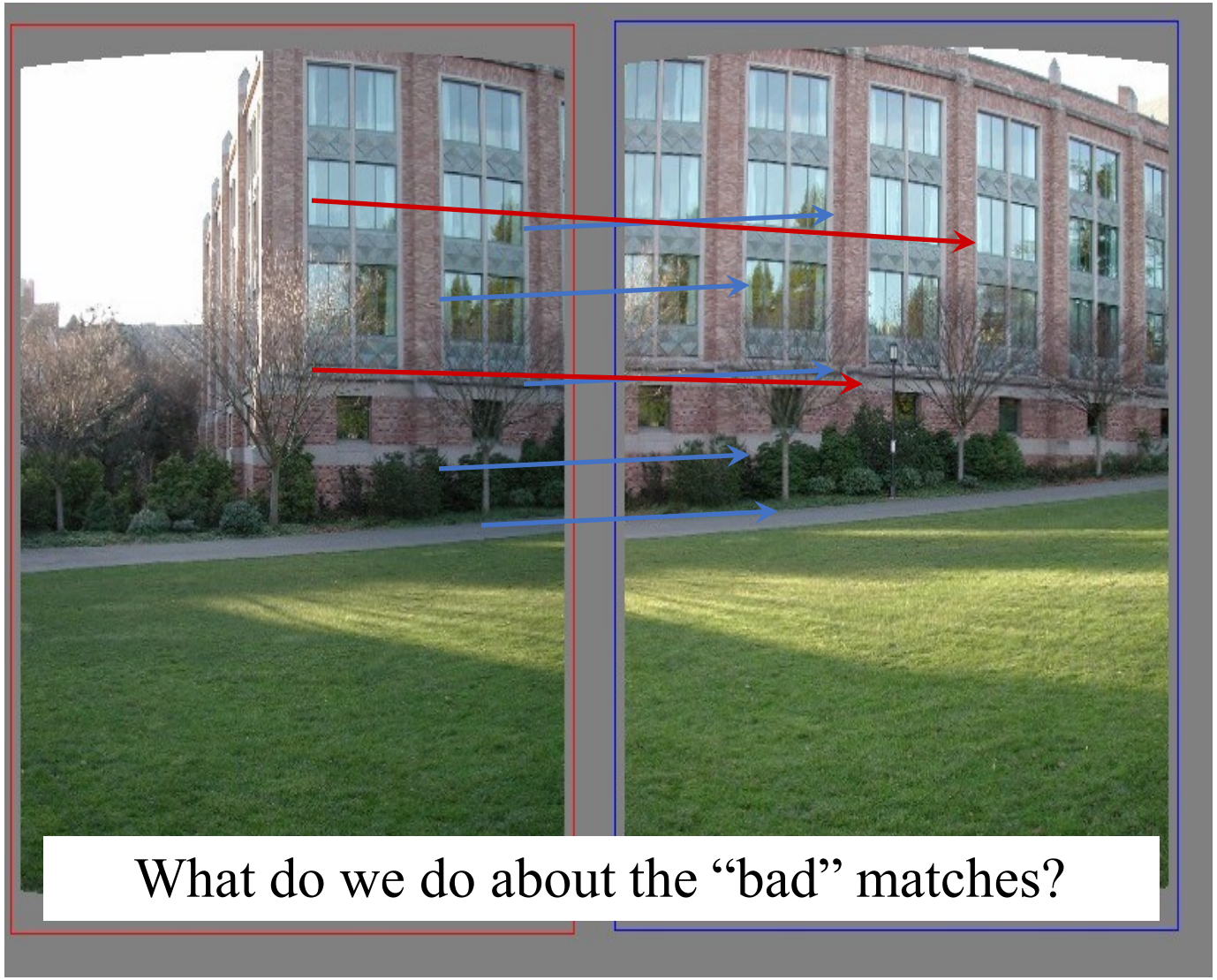


Lead to next class on RANSAC

Today’s class

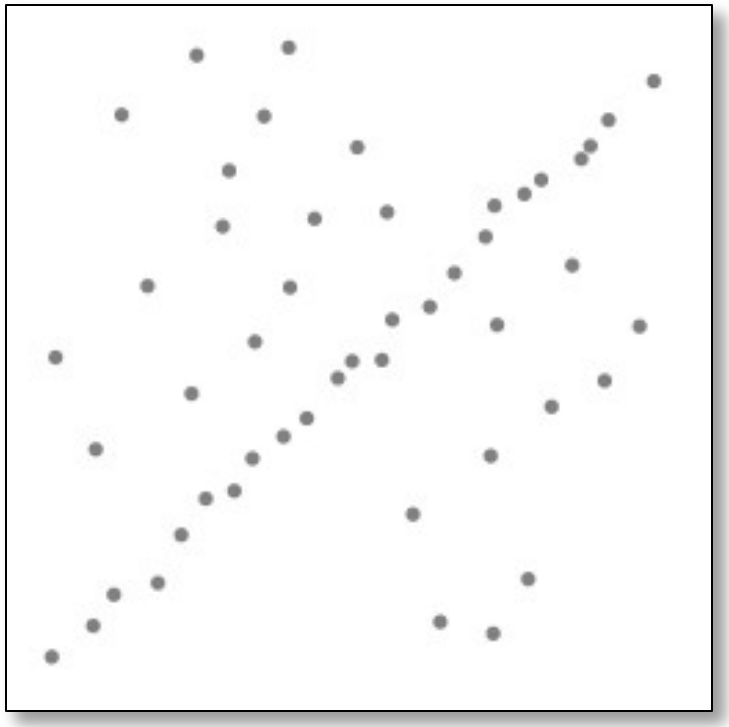
- Fitting with outliers – RANSAC
- Warping
- Blending

Matching features

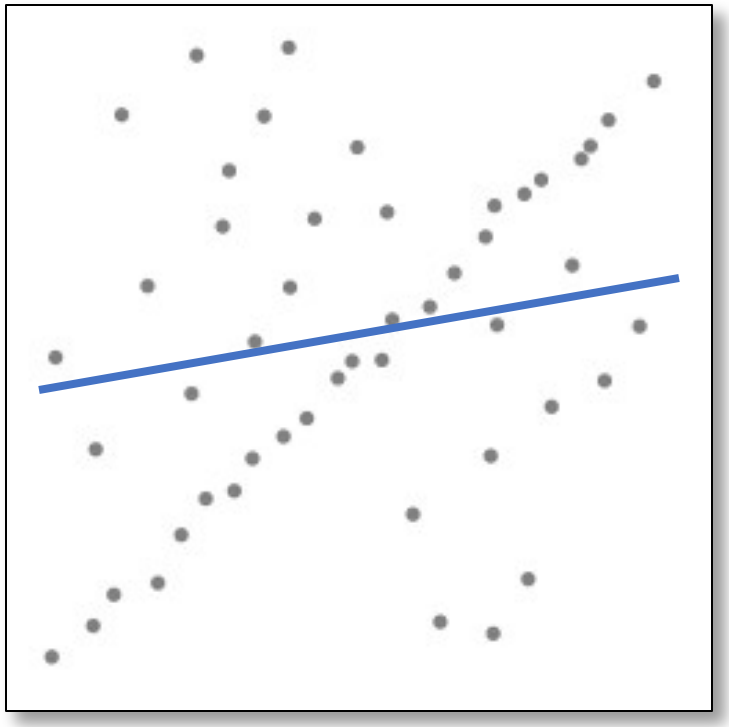


Robustness

- Let’s consider the problem of linear regression



Problem: Fit a line to these datapoints



Least squares fit

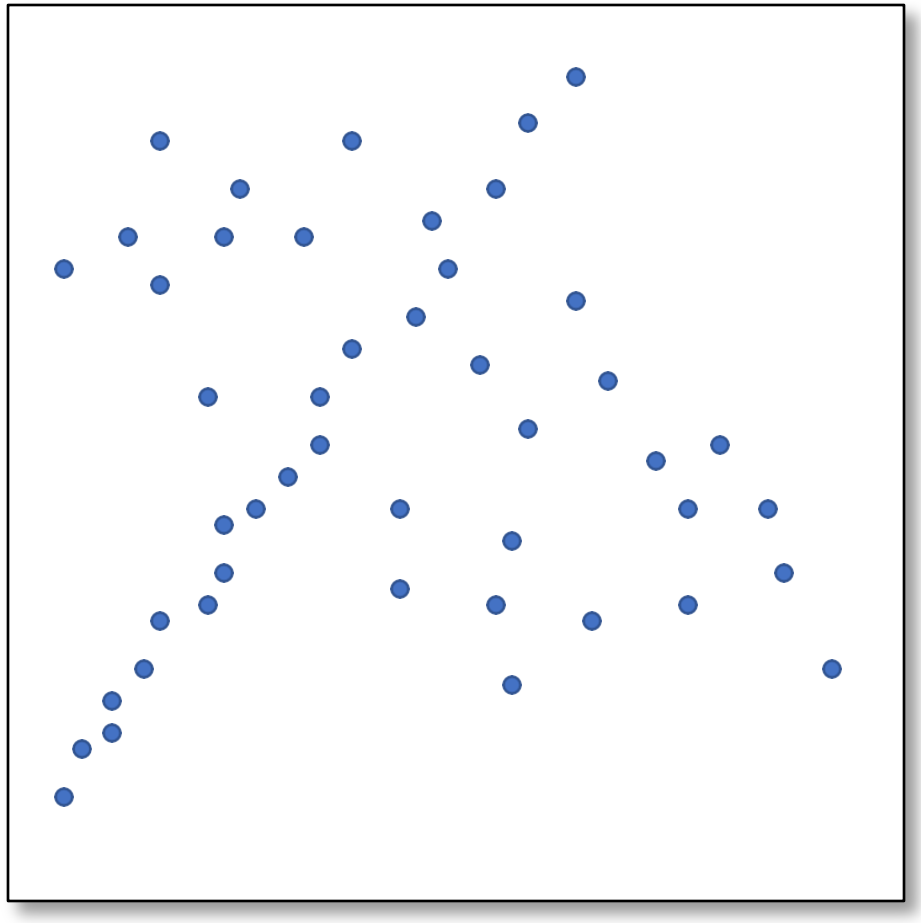
- How can we fix this?

Idea

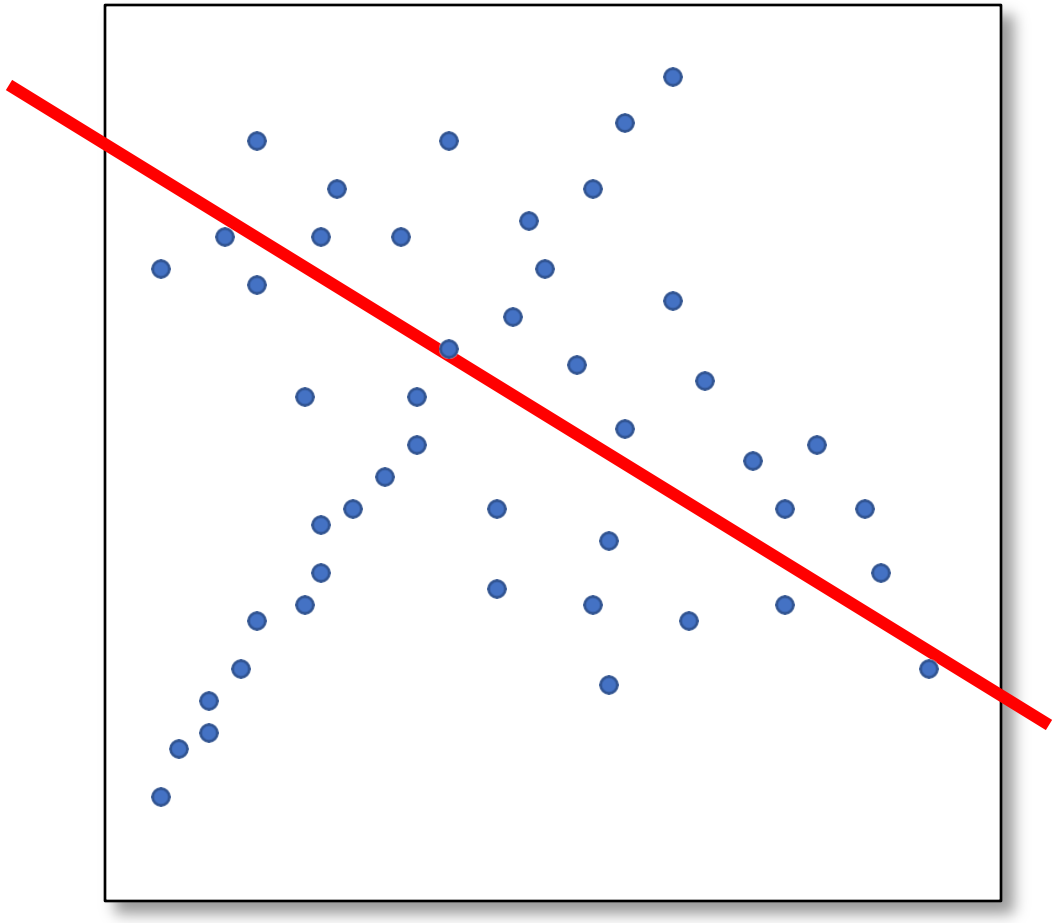
- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

Counting inliers

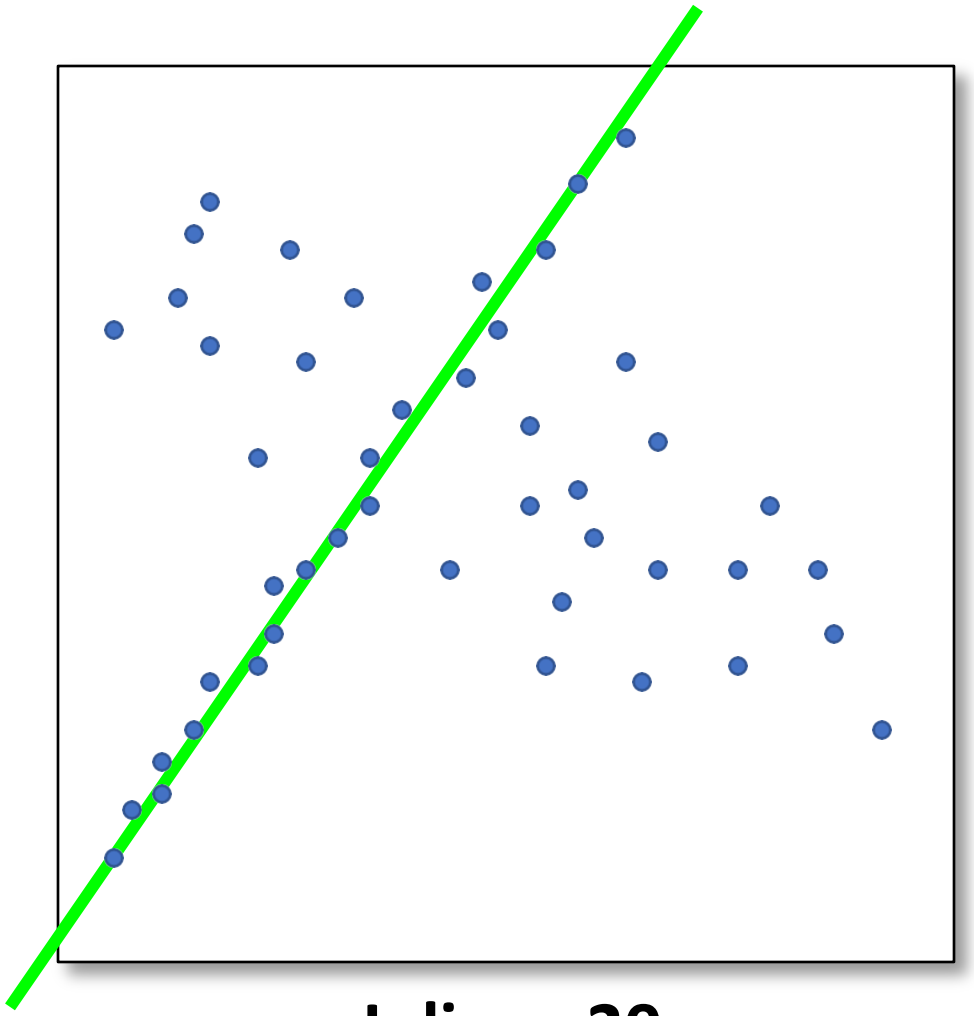


Counting inliers



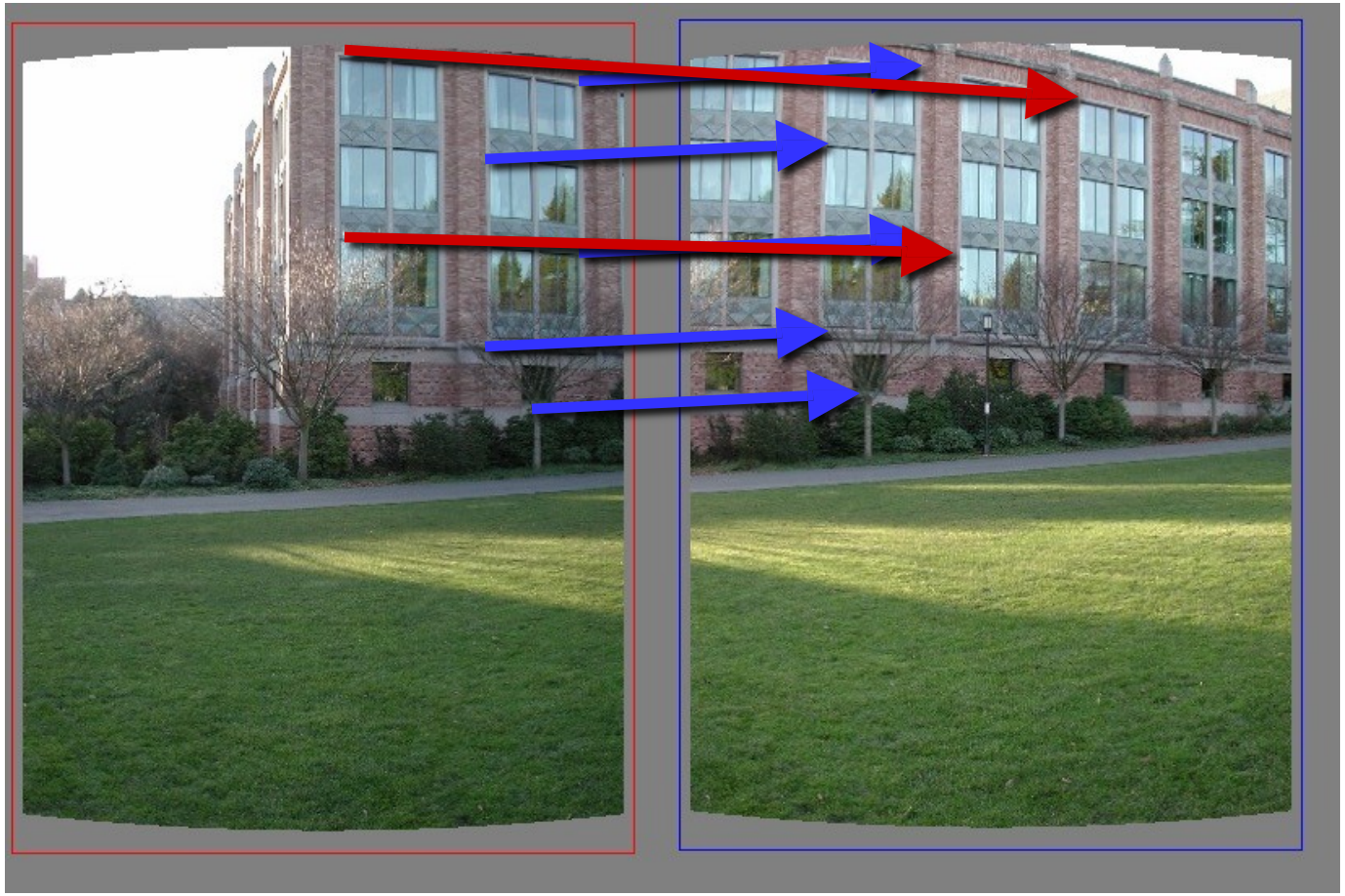
Inliers: 3

Counting inliers

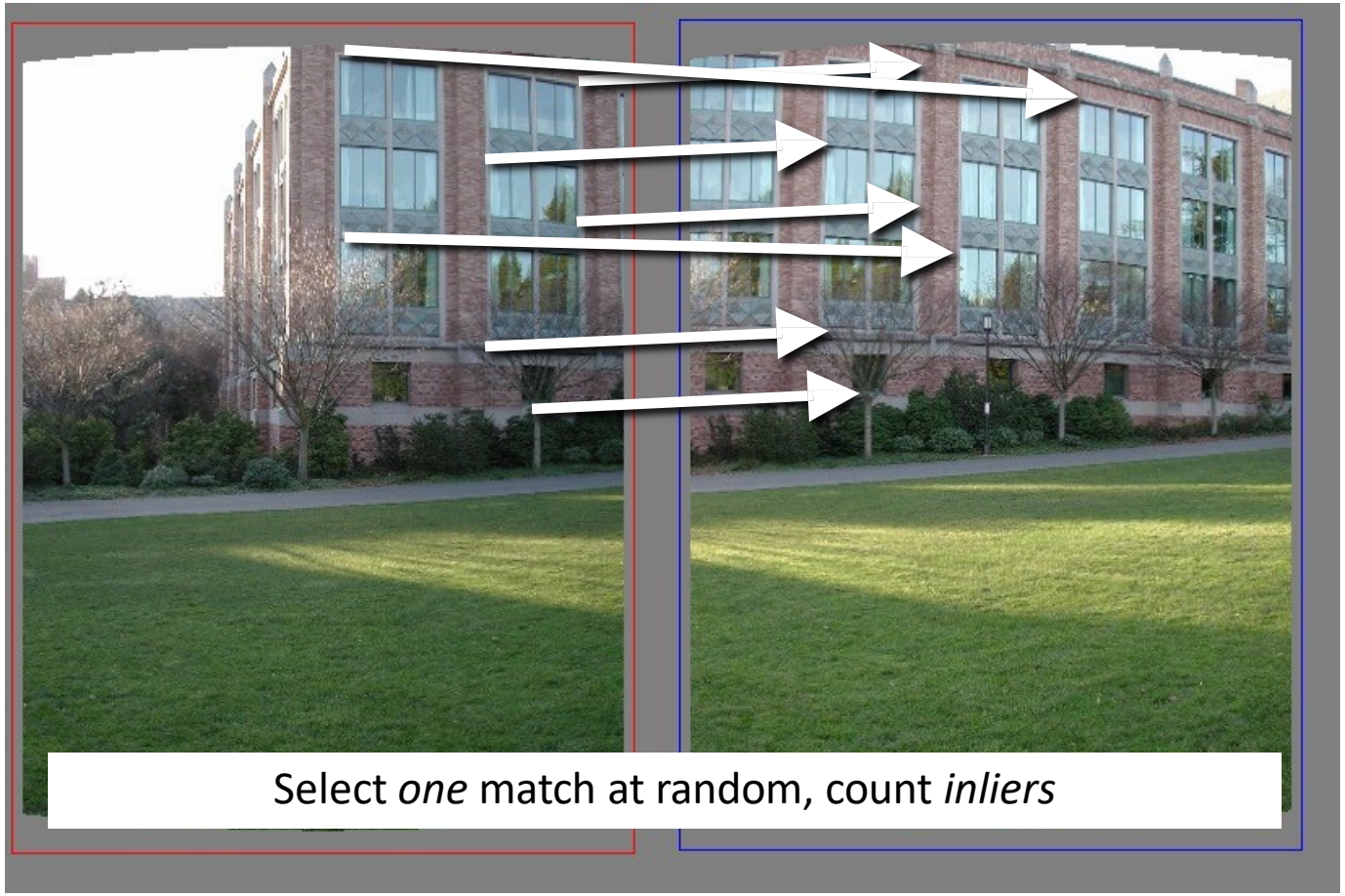


Inliers: 20

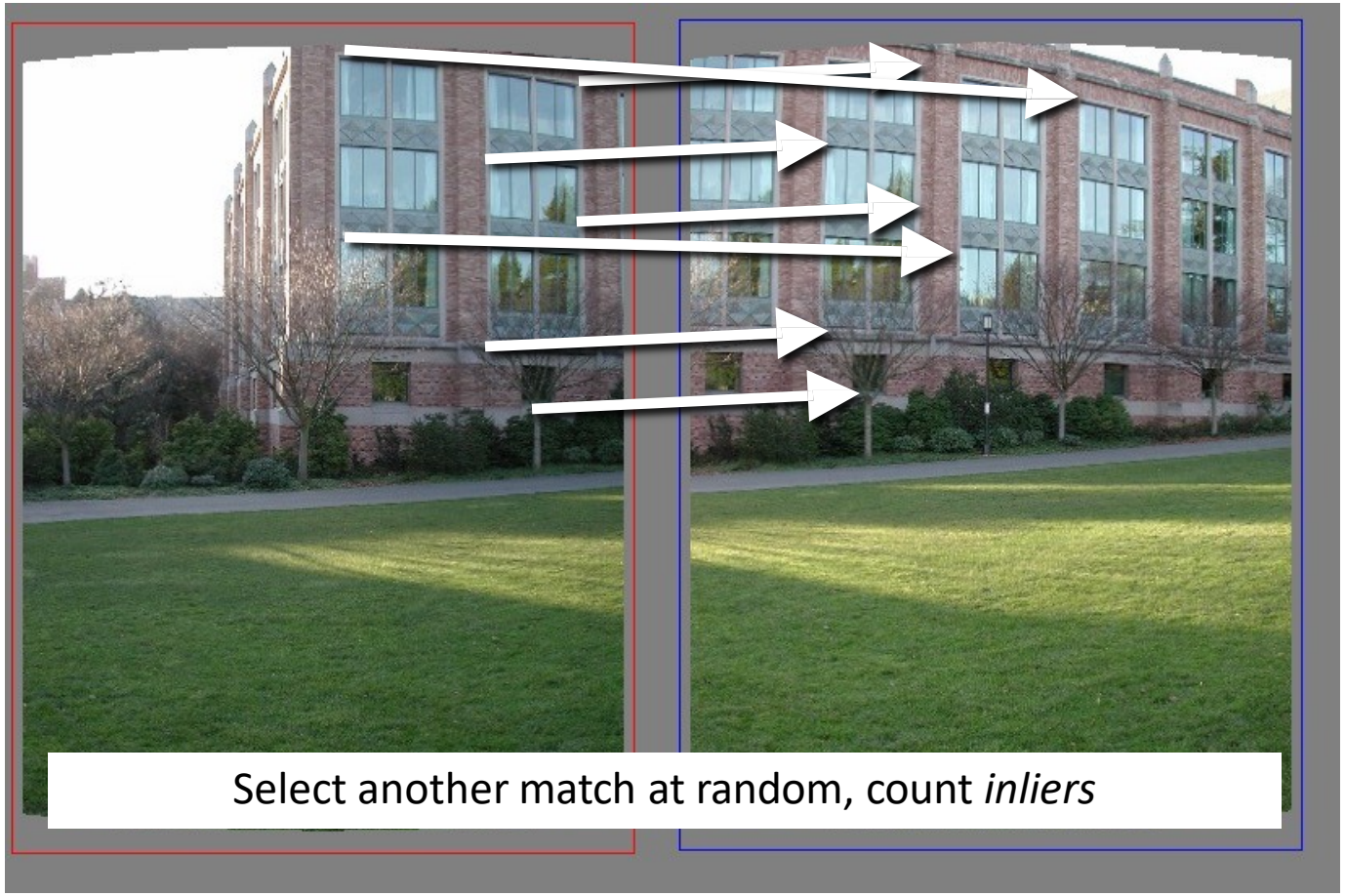
Translations



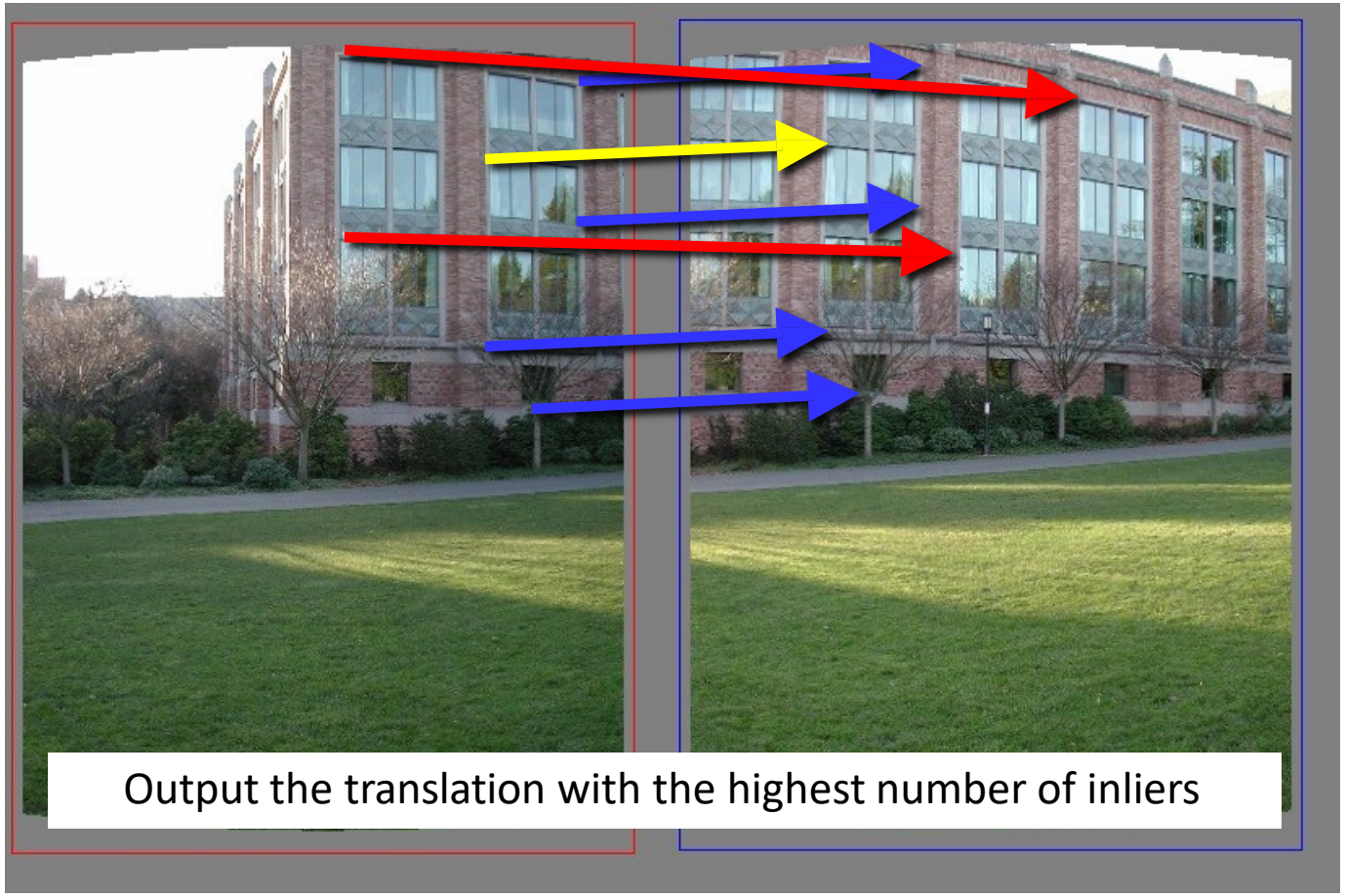
RAndom SAmple Consensus



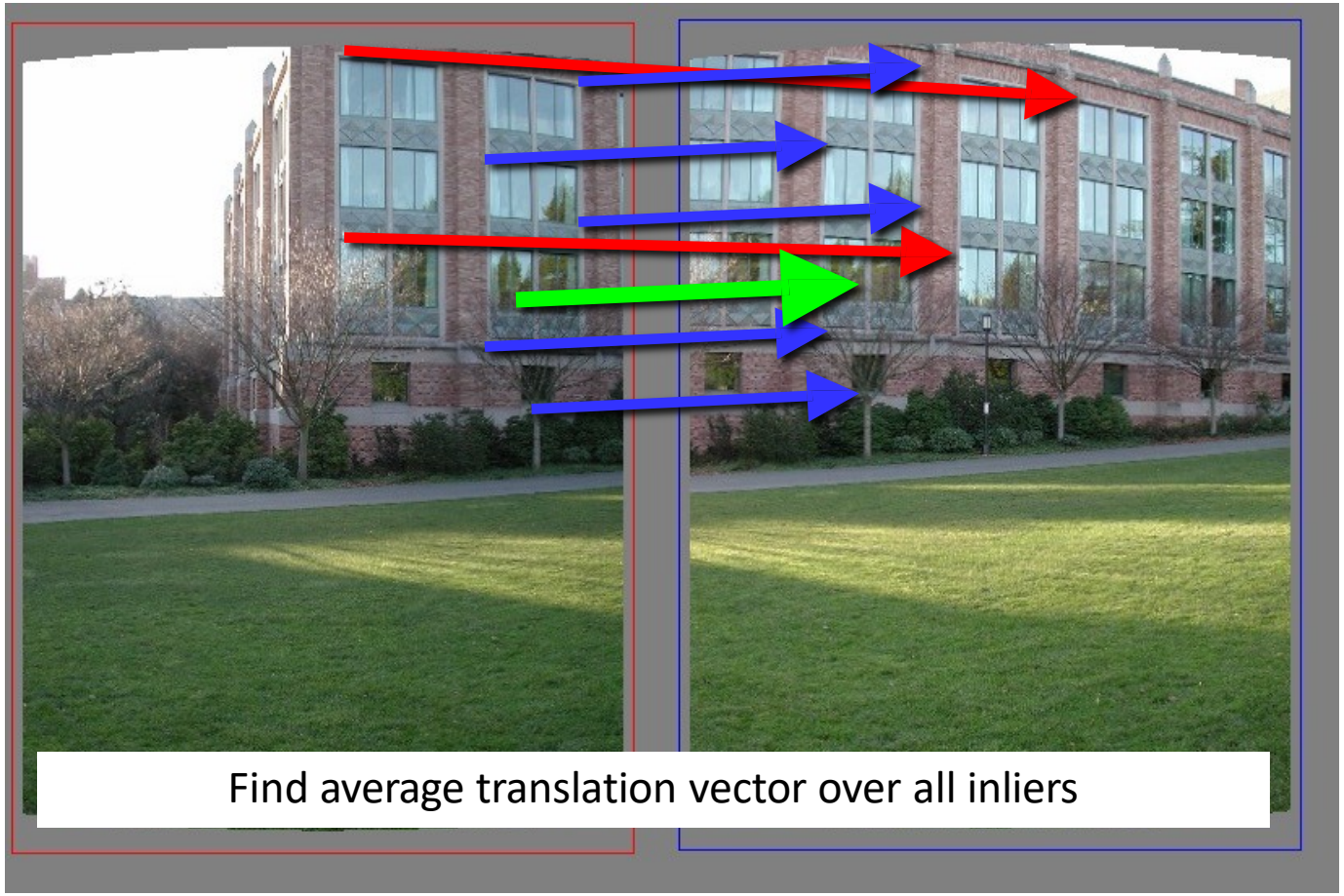
RAndom SAmple Consensus



RAndom SAmple Consensus



Final step: least squares fit



RANSAC

- Idea:
  - All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
    - RANSAC only has guarantees if there are < 50% outliers
  - “All good matches are alike; every bad match is bad in its own way.”

    – Tolstoy via Alyosha Efros

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set of inliers

RANSAC for estimating homography

RANSAC loop:

1. **Select four feature pairs (at random)**
2. **Compute homography H (exact)**
3. **Compute *inliers* where $dist(p_i', H\,p_i) < \varepsilon$**
4. **Keep largest set of inliers**
5. **Re-compute least-squares H estimate on all of the inliers**

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. **Repeat *N* times**
  5. **Choose the model that has the largest set of inliers**

How many rounds?

- If we have to choose *s* samples each time
  - with an outlier ratio *e*
  - and we want the right answer with probability *p*

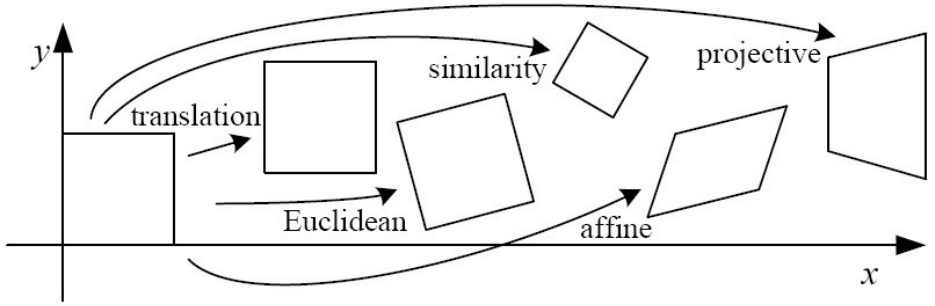
$$N \geq \frac{\log(1-p)}{\log(1-(1-e)^s)}$$

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

p = 0.99

How big is *s*?

- For alignment, depends on the motion model
  - Here, each sample is a correspondence (pair of matching points)



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

RANSAC pros and cons

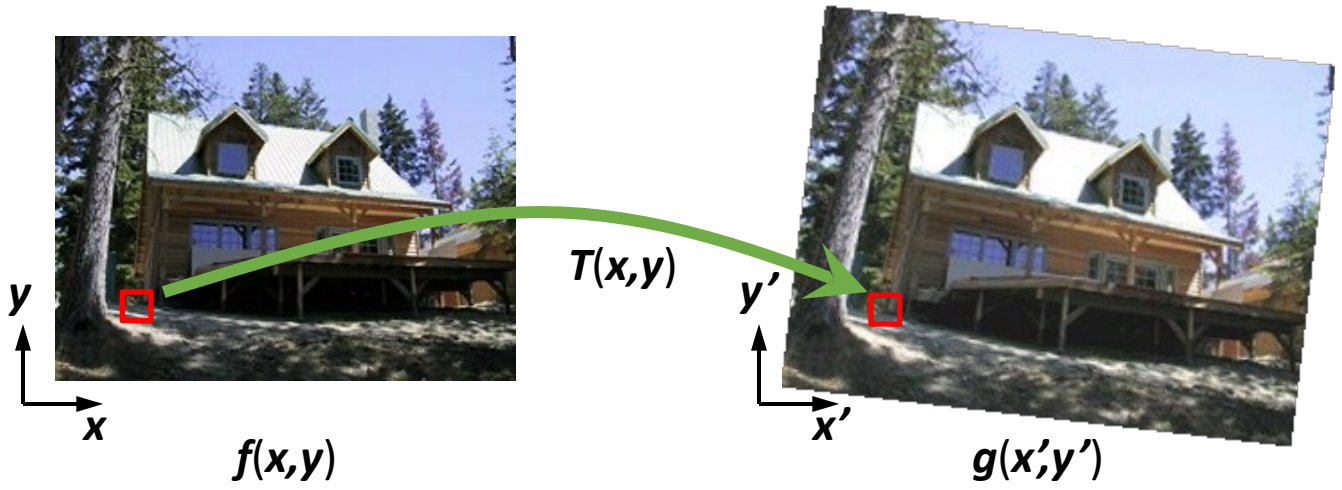
- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Parameters to tune
  - Sometimes too many iterations are required
  - Can fail for extremely low inlier ratios
  - We can often do better than brute-force sampling

Today’s class

- Fitting with outliers – RANSAC
- Warping
- Blending

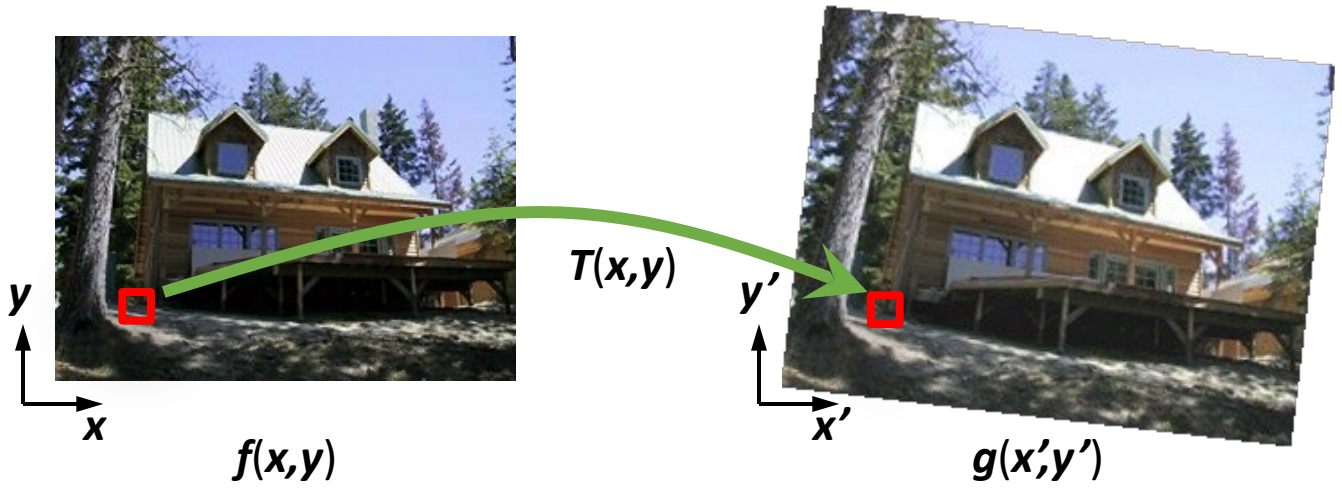
Implementing image warping

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



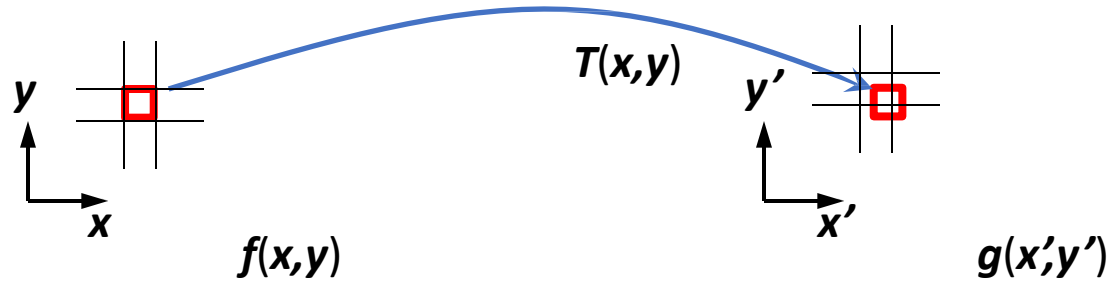
Forward Warping

- Send each pixel (***x,y***) to its corresponding location (***x’,y’***) = ***T***(***x,y***) in ***g***(***x’,y’***)
    - What if pixel lands “between” two pixels?



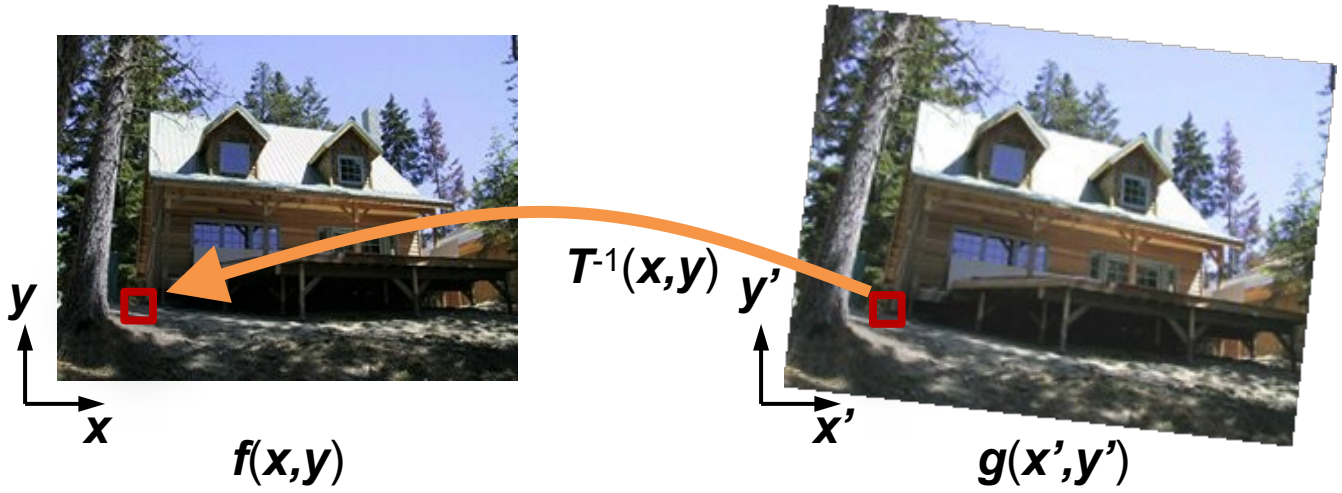
Forward Warping

- Send each pixel (***x,y***) to its corresponding location (***x’,y’***) = ***T***(***x,y***) in ***g***(***x’,y’***)
    - What if pixel lands “between” two pixels?
    - Answer: add “contribution” to several pixels, normalize later (*splatting*)
    - Can still result in holes

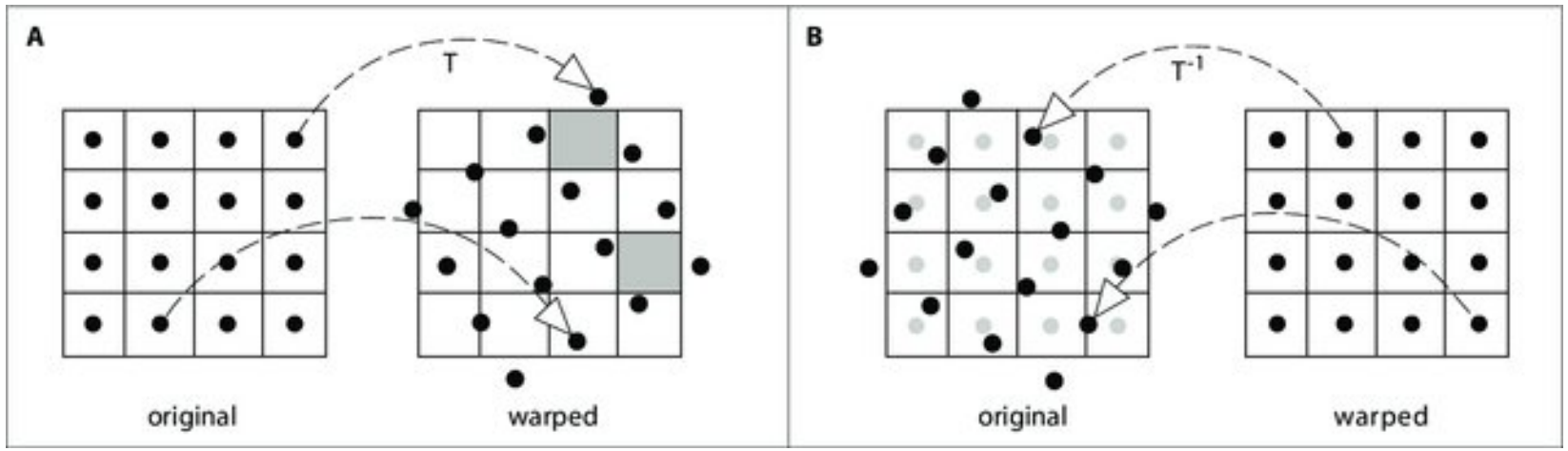


Inverse Warping

- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x,y)$ in $f(x,y)$
  - Requires taking the inverse of the transform
  - What if pixel comes from "between" two pixels?

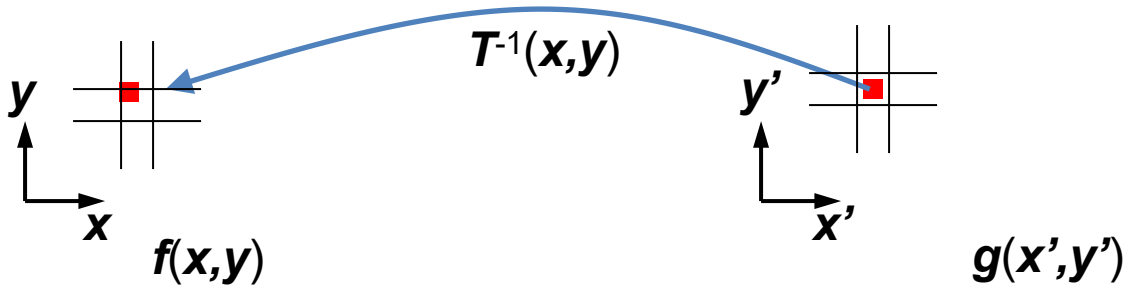


The idea



Inverse Warping

- Get each pixel ***g***(***x'***) from its corresponding location ***x'*** = ***h***(***x***) in ***f***(***x***)
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic
  - sinc
- Needed to prevent artifacts

Today’s class

- Fitting with outliers – RANSAC
- Warping
- Blending

Jia-Bin Huang
@jbhuang0604

The ControlNet illusion art is FUN!

In some sense, it's a *Hybrid Image* (17-year-old method). It was my first homework in the computer vision class. I remember that it takes time to properly align the two images to get good results.

Now it's only one sentence and one click!

https://stable-diffusion-art.com/controlnet/

1:46 PM · Sep 27, 2023 · **821** Views

https://stable-diffusion-art.com/controlnet/

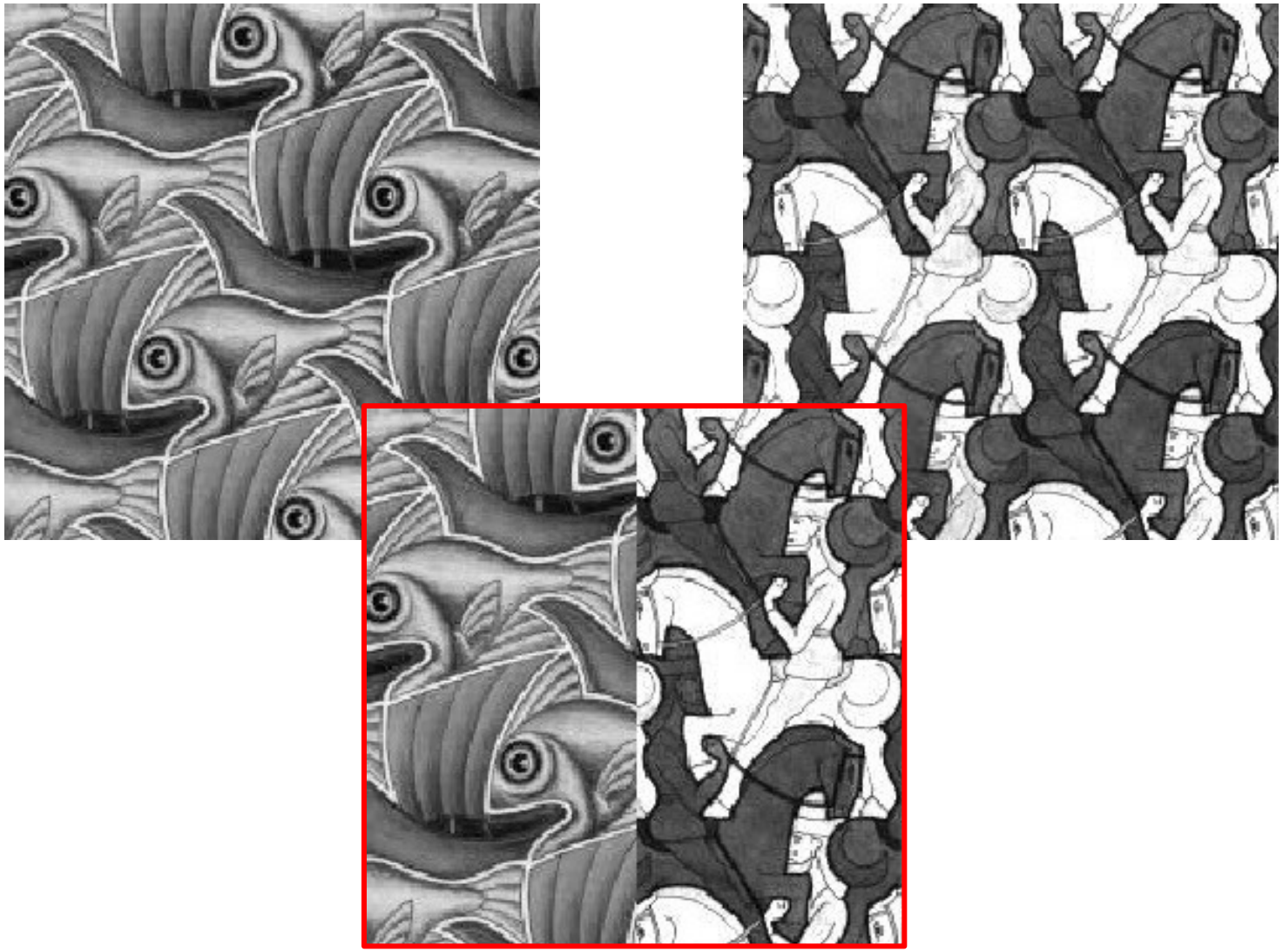
Blending

- We’ve aligned the images – now what?

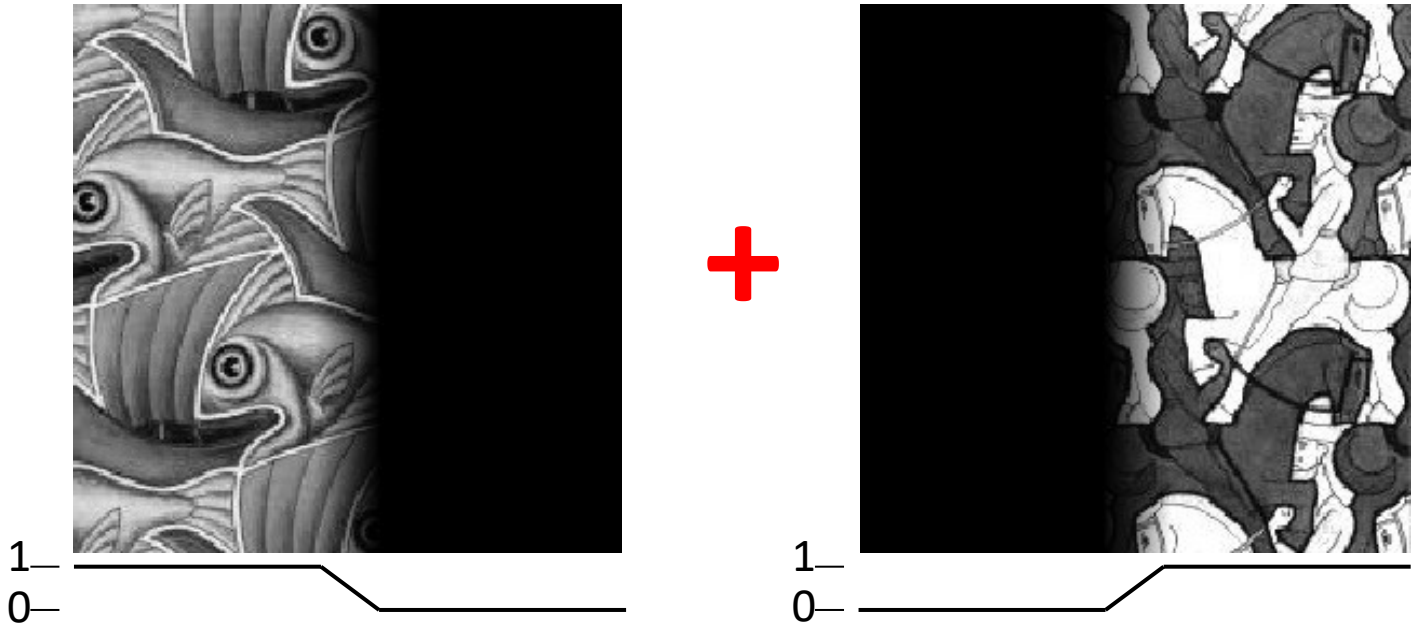
Blending

- Want to seamlessly blend them together

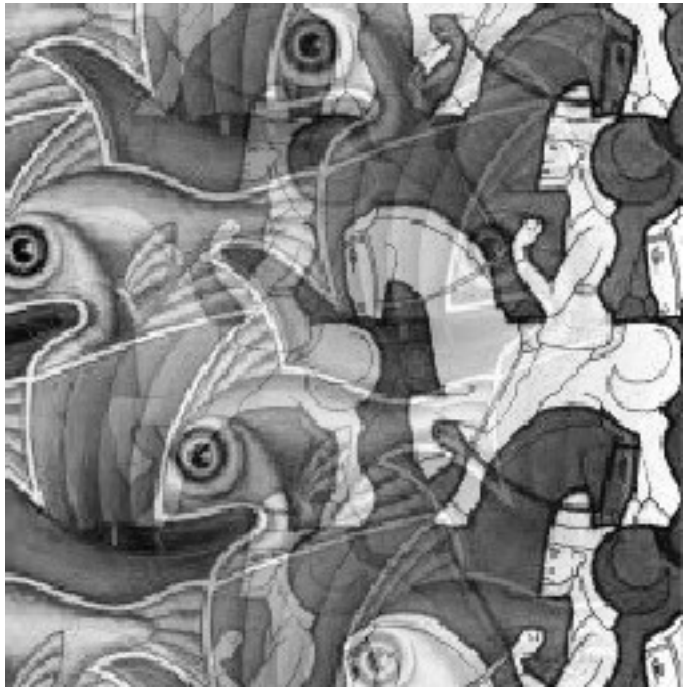
Image Blending

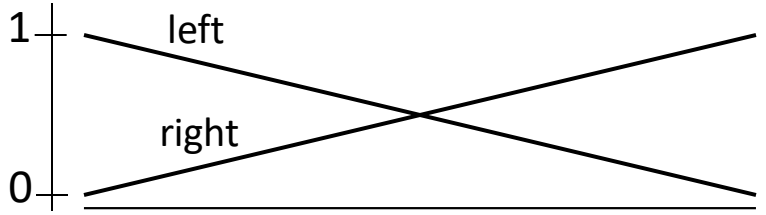


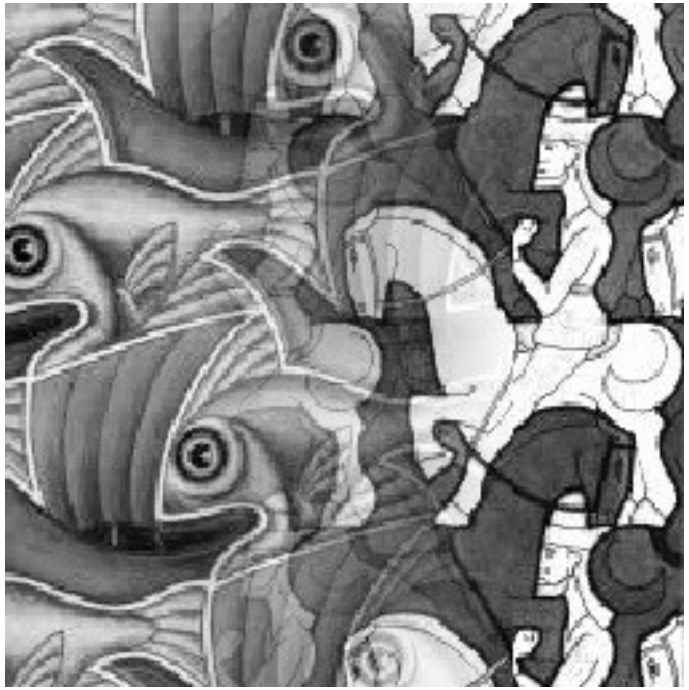
Feathering

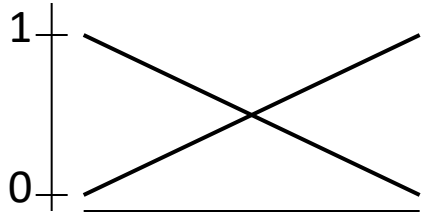


Effect of window size

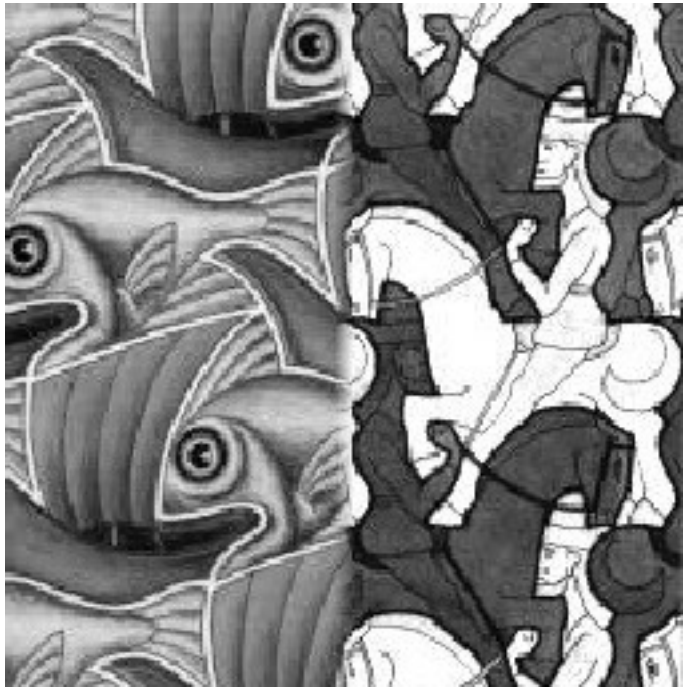


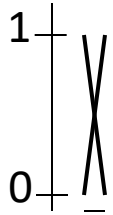


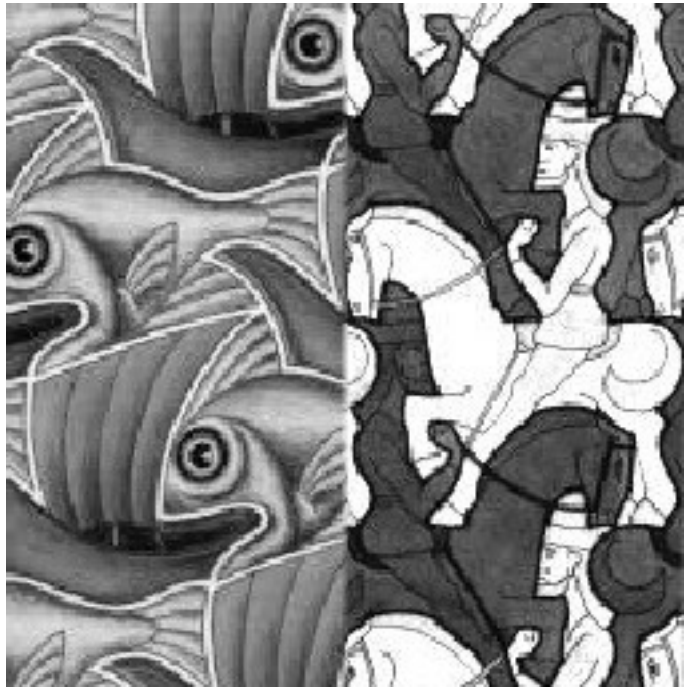


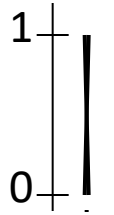


Effect of window size

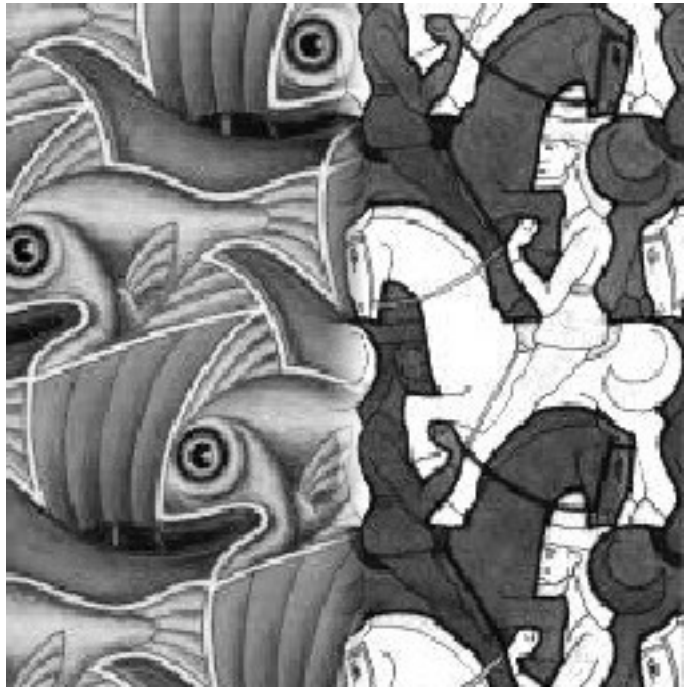








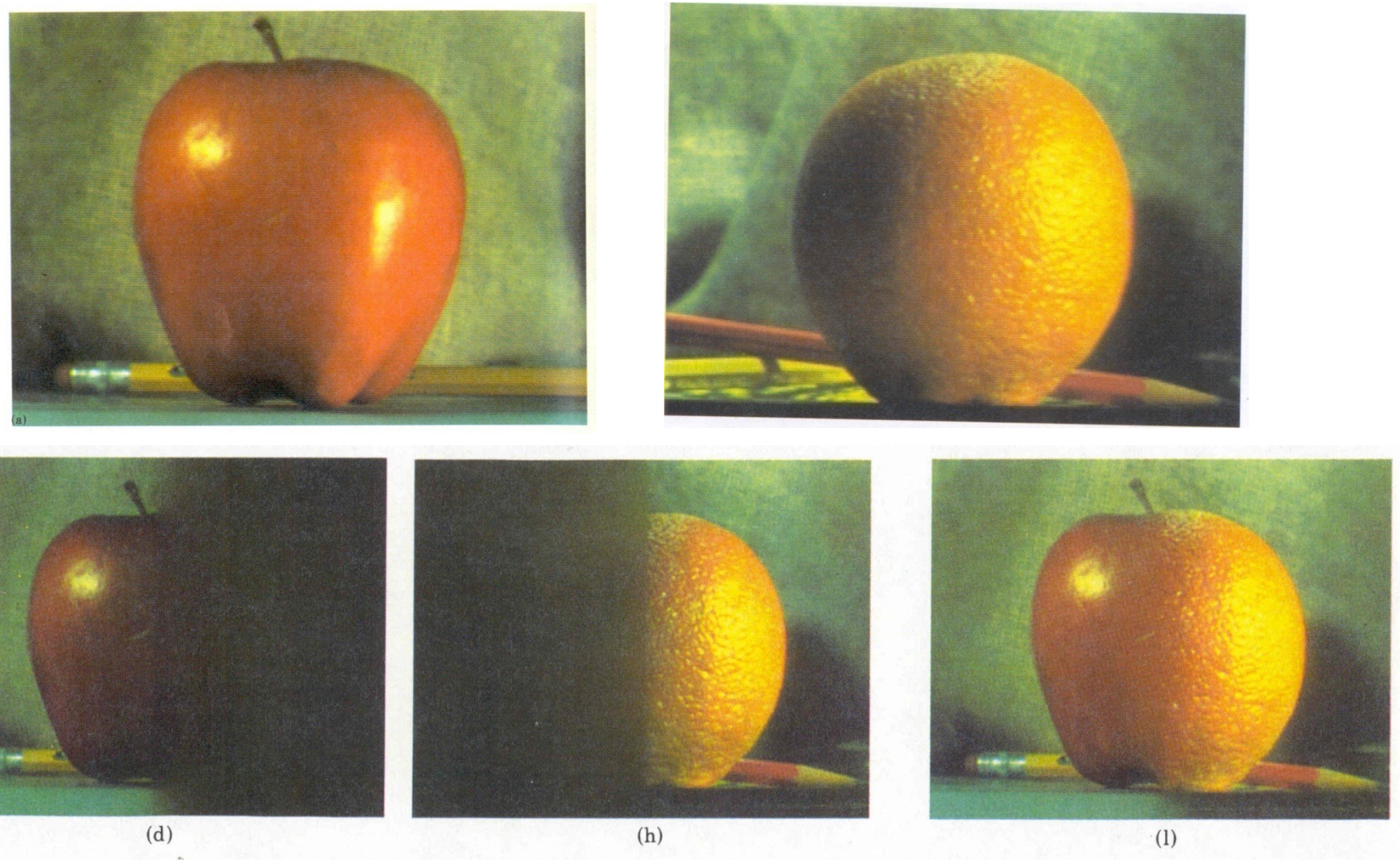
Good window size



“Optimal” window: smooth but not ghosted

- Doesn’t always work...

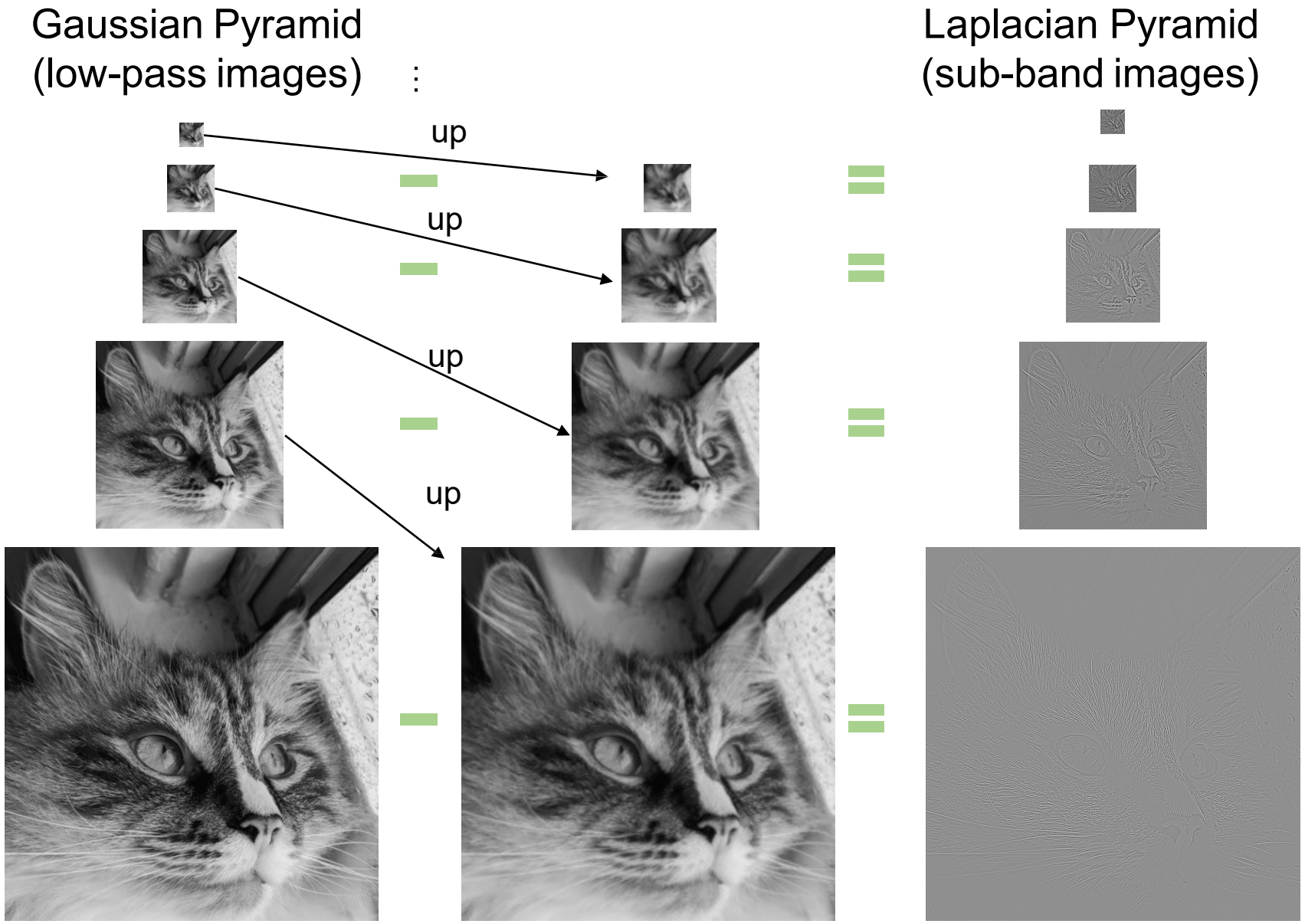
Pyramid blending



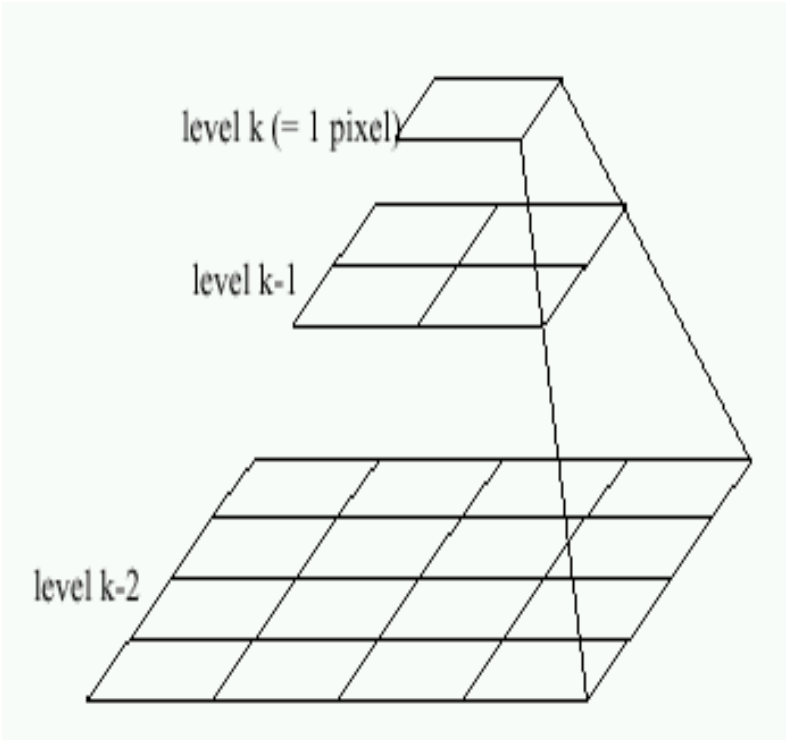
Create a Laplacian pyramid, blend each level

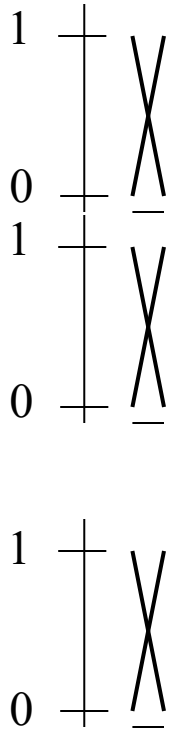
- Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

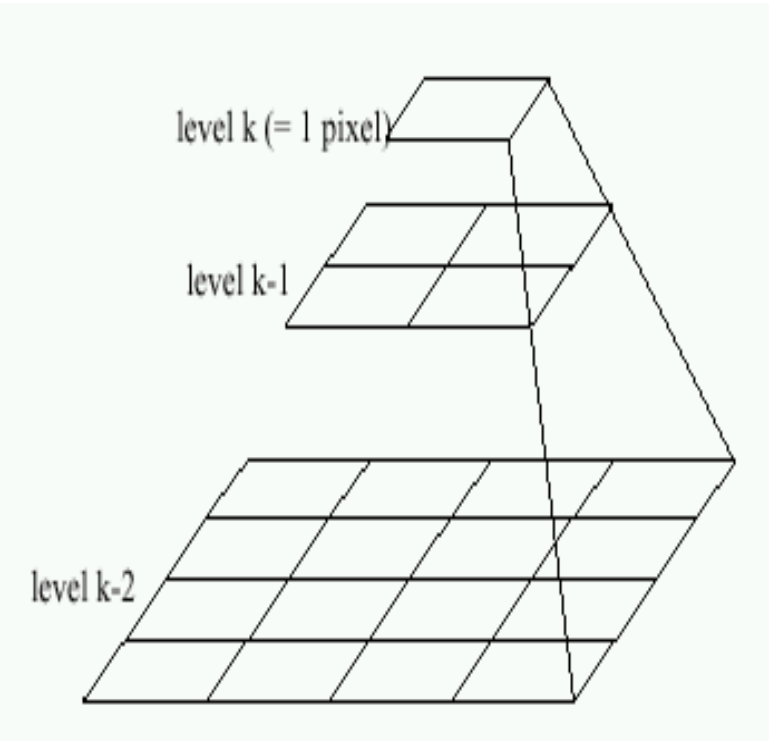
Band-pass filtering in spatial domain



Pyramid Blending



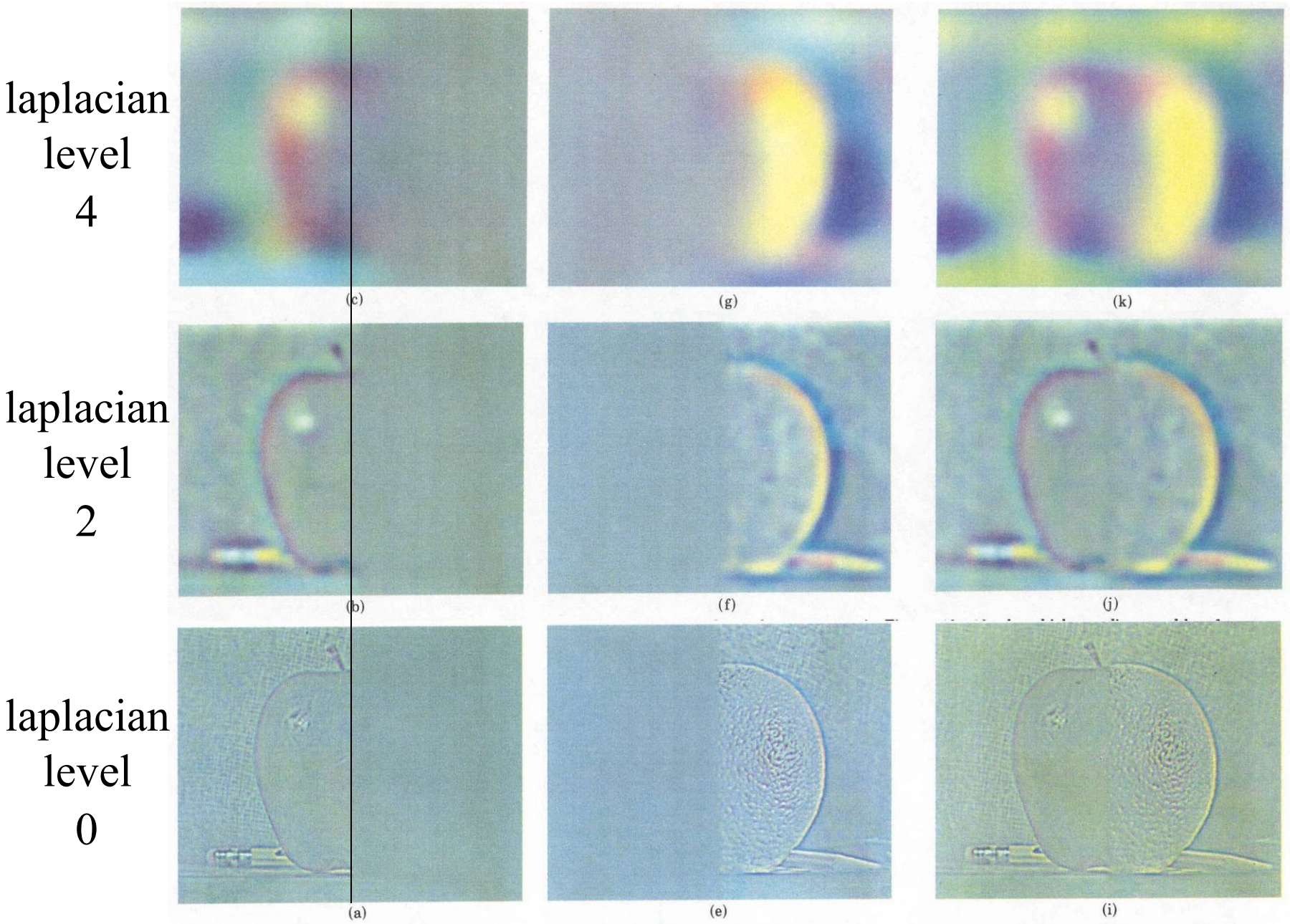




Left pyramid

blend

Right pyramid

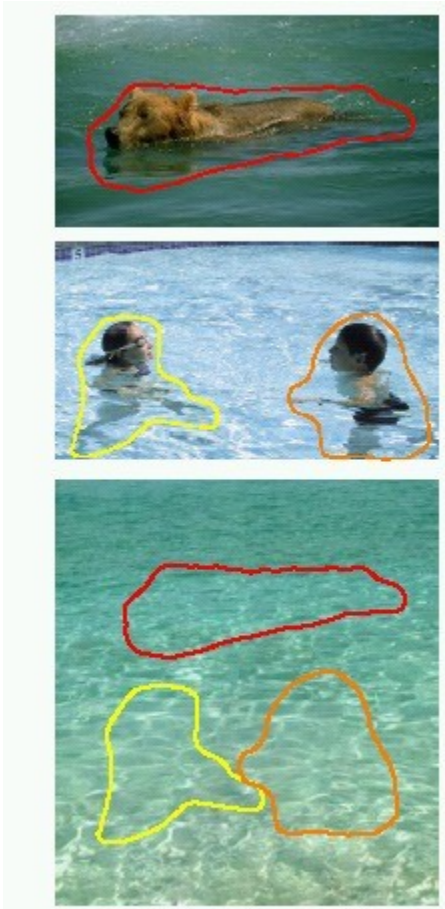

laplacian level 4
laplacian level 2
laplacian level 0
(c)
(g)
(k)
(b)
(f)
(j)
(a)
(e)
(i)
left pyramid
right pyramid
blended pyramid

Laplacian Pyramid: Blending

- General Approach:
  1. Build Laplacian pyramids *LA* and *LB* from images *A* and *B*
  2. Build a Gaussian pyramid *GR* from selected region *R*
  3. Form a combined pyramid *LS* from *LA* and *LB* using nodes of *GR* as weights:
     - *LS(i,j) = GR(I,j,)*LA(I,j) + (1-GR(I,j))*LB(I,j)*
  4. Collapse the *LS* pyramid to get the final blended image

Poisson Image Editing

For more info: [Perez et al, SIGGRAPH 2003]



sources/destinations

cloning

seamless cloning

Solve a partial differential equation known as the Poisson equation to ensure that the gradients (or differences) between corresponding pixels in the source and destination images are minimized within a certain region of interest

Fun with homographies

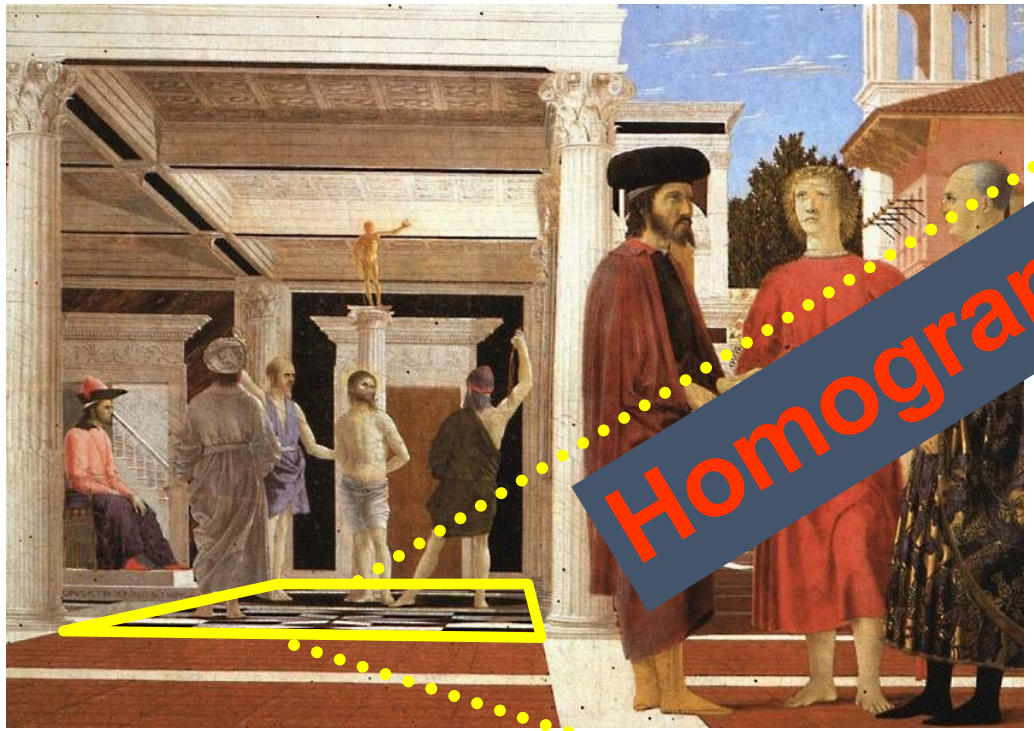
Original image

St.Petersburg
photo by A. Tikhonov

Virtual camera rotations

Analysing patterns and shapes

What is the shape of the b/w floor pattern?



Homography

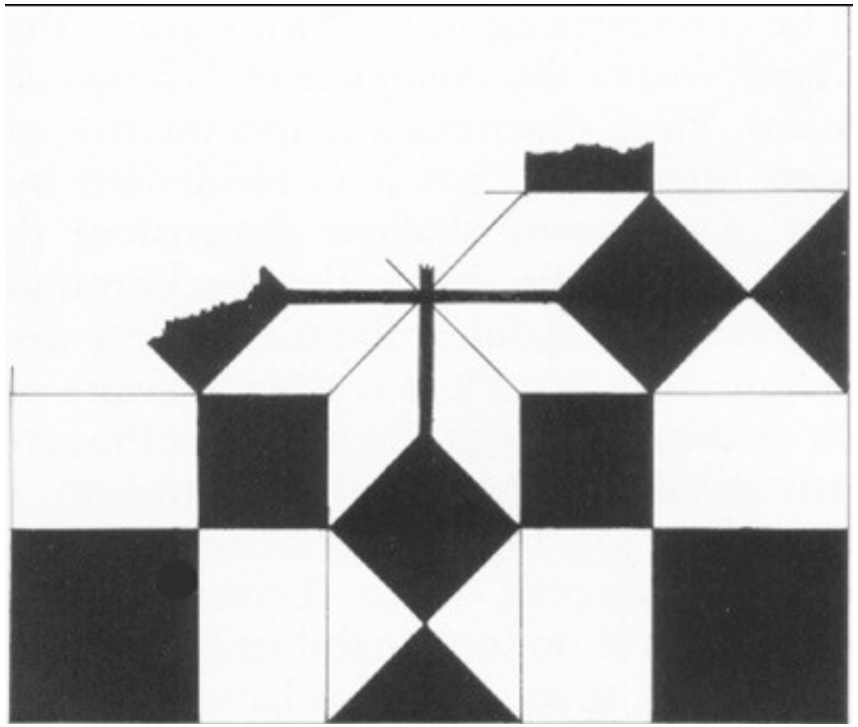
The floor (enlarged)

Automatically rectified floor

Slide from Criminisi

Analysing patterns and shapes

Automatic rectification

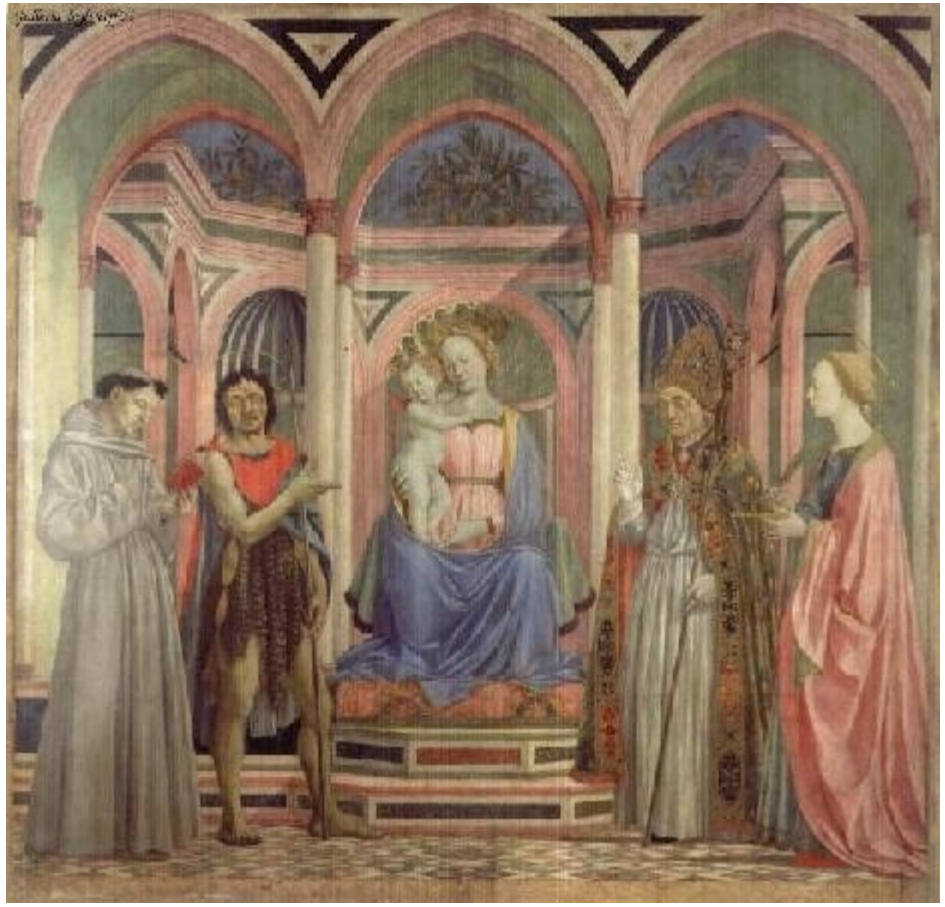


From Martin Kemp *The Science of Art*
(manual reconstruction)

2 patterns have been discovered !

Slide from Criminisi

Analysing patterns and shapes



St. Lucy Altarpiece, D. Veneziano

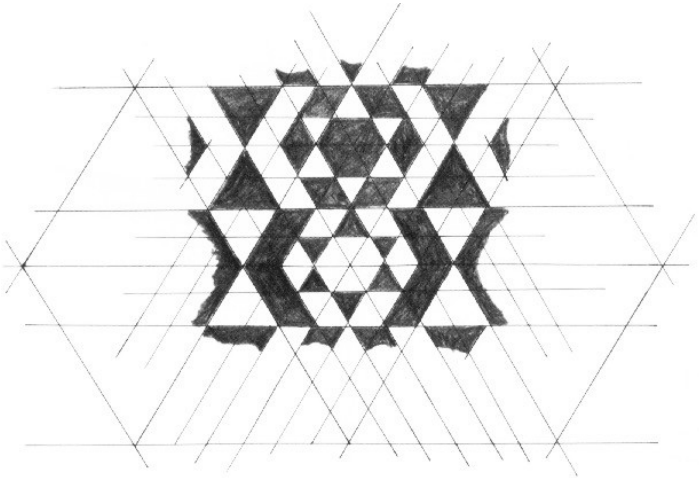
Slide from Criminisi

What is the (complicated) shape of the floor pattern?

Automatically rectified floor

Analysing patterns and shapes

Automatic rectification



From Martin Kemp, *The Science of Art* *(manual reconstruction)*

Slide from Criminisi

virtual wide-angle camera

Some panorama examples

- Every image on Google Streetview

Slide Credits

- CS5670, Introduction to Computer Vision, Cornell Tech, by Noah Snavely.
- CS 194-26/294-26: Intro to Computer Vision and Computational Photography, UC Berkeley, by Alyosha Efros.
- Fall 2022 CS 543/ECE 549: Computer Vision, UIUC, by Svetlana Lazebnik.

Acknowledgements: some slides and material from Bernt Schiele, Mario Fritz, Michael Black, Bill Freeman, Fei-Fei Li, Justin Johnson, Serena Yeung, R. Szelisky, Ioannis Gkioulekas, Roni Sengupta, Andreas Geiger