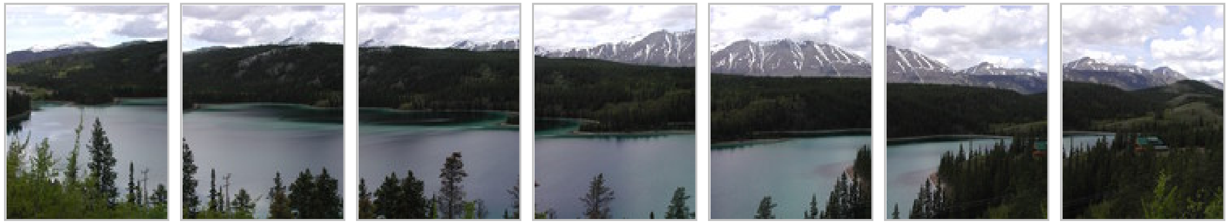
Image Stitching

Linda Shapiro
CSE 455

- Combine two or more overlapping images to make one larger image



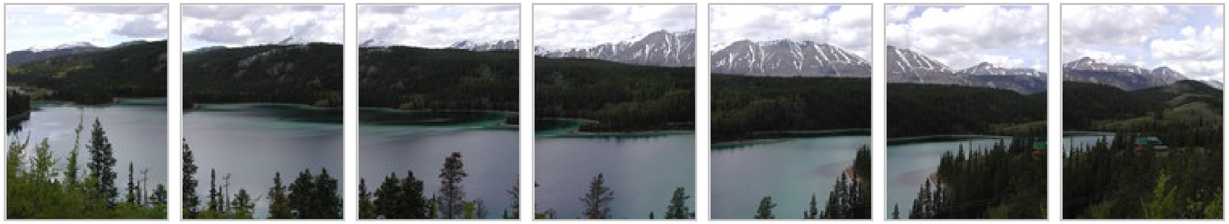
Slide credit: [Vaibhav Vaish]

How to do it?

- Basic Procedure
  1. Take a sequence of images from the same position
     1. Rotate the camera about its optical center
  2. Compute transformation between second image and first
  3. Shift the second image to overlap with the first
  4. Blend the two together to create a mosaic
  5. If there are more images, repeat

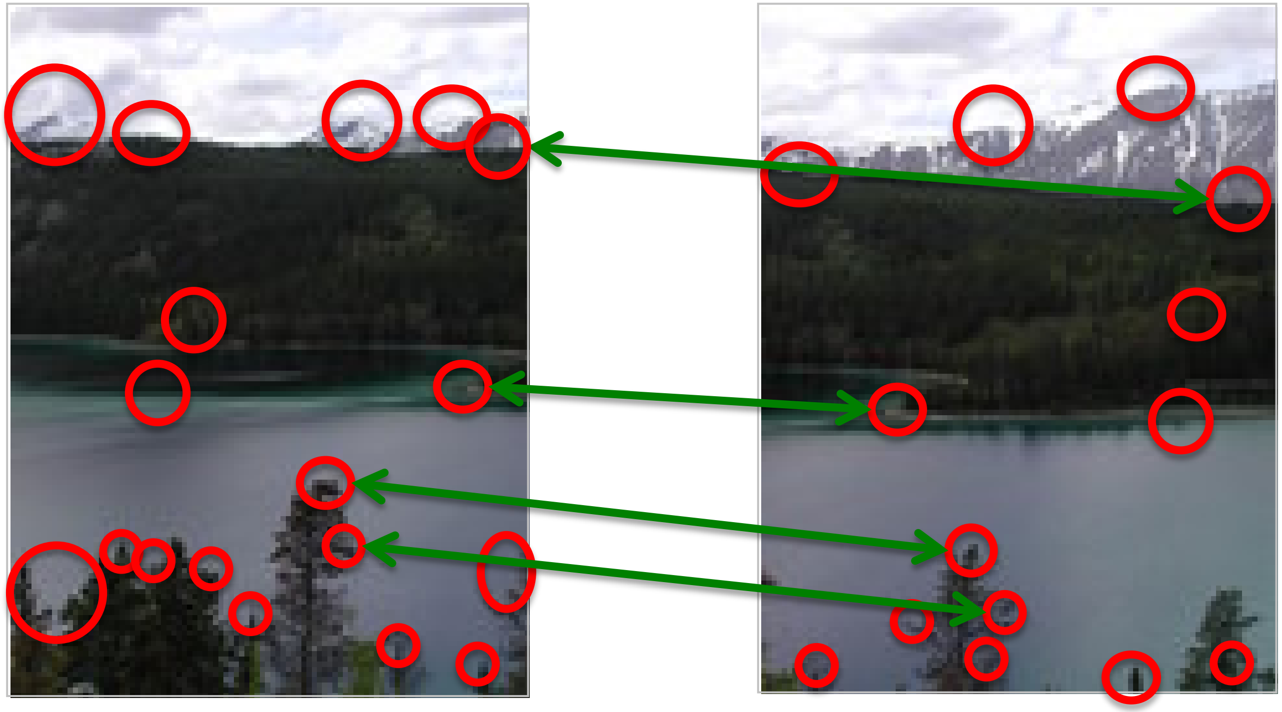
1. Take a sequence of images from the same position

- Rotate the camera about its optical center

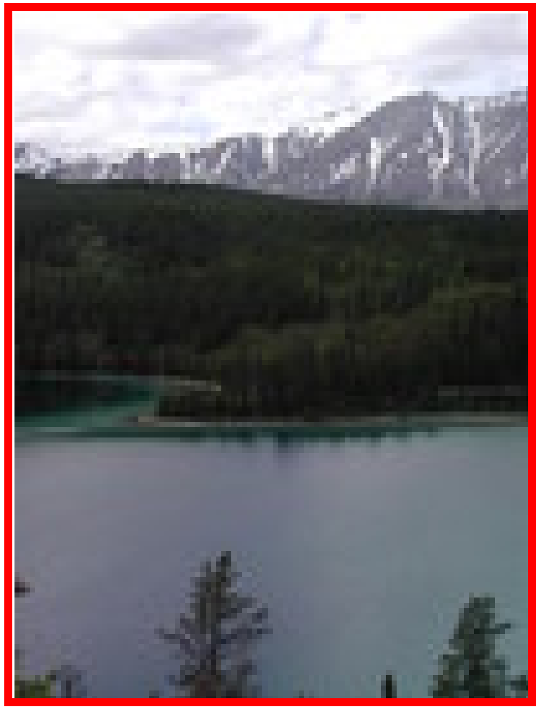


2. Compute transformation between images

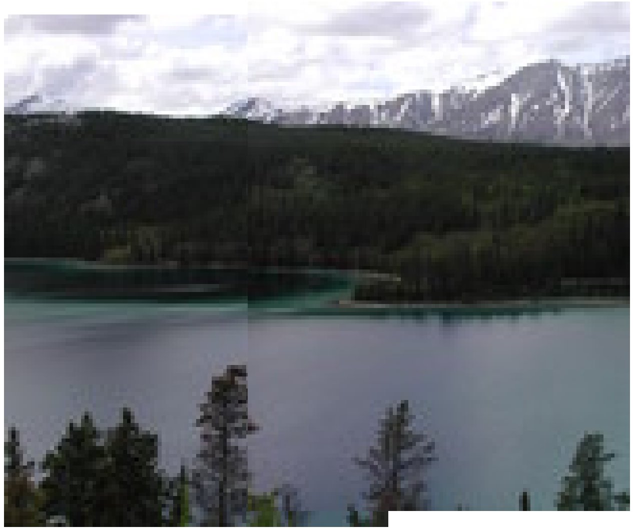
- Extract interest points
- Find Matches
- Compute transformation ?



3. Shift the images to overlap



4. Blend the two together to create a mosaic



5. Repeat for all images

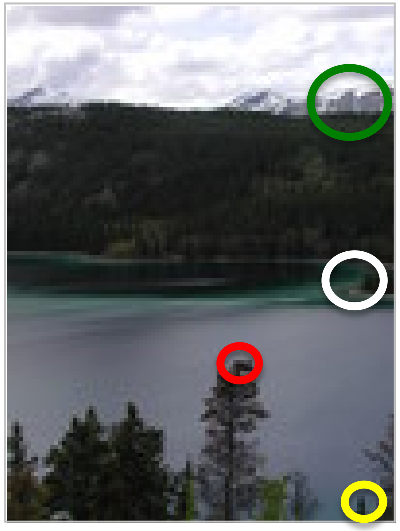
How to do it?

- Basic Procedure
  1. Take a sequence of images from the same position
     1. Rotate the camera about its optical center
  2. Compute transformation between second image and first
  3. Shift the second image to overlap with the first
  4. Blend the two together to create a mosaic
  5. If there are more images, repeat

Compute Transformations

- ✓ Extract interest points
- ✓ Find good matches
- Compute transformation

Let's assume we are given a set of good matching interest points



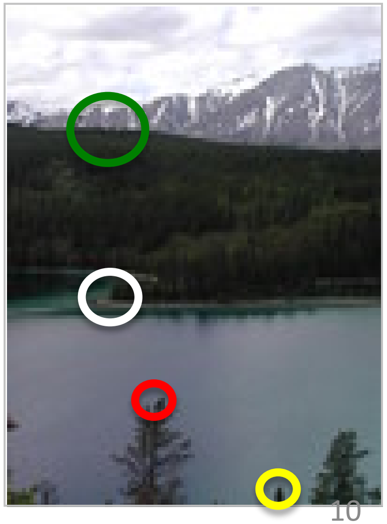
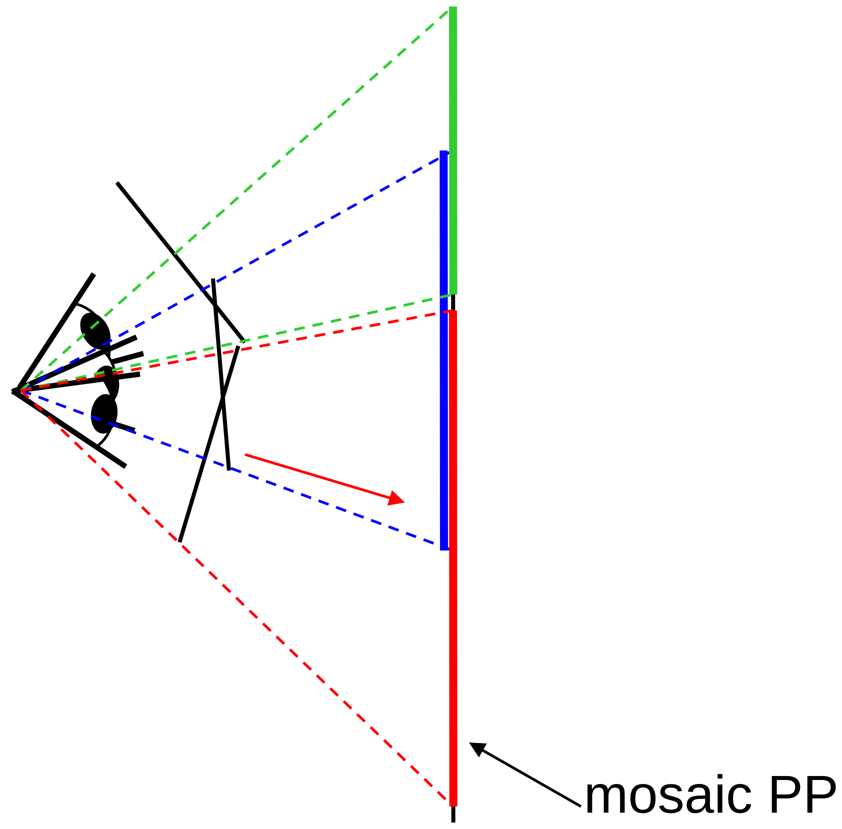


Image reprojection



- The mosaic has a natural interpretation in 3D
  - The images are reprojected onto a common plane
  - The mosaic is formed on this plane

Example

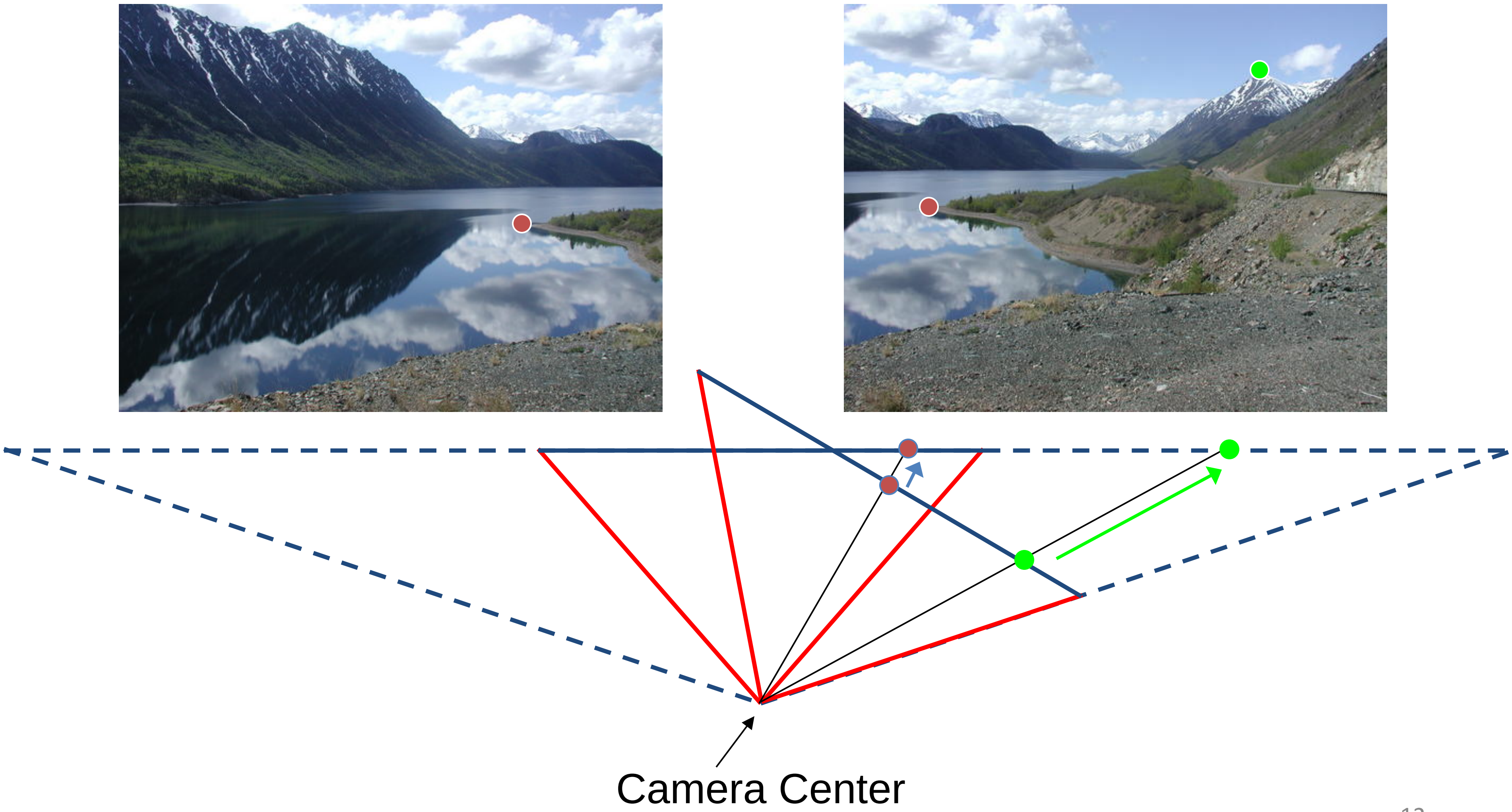
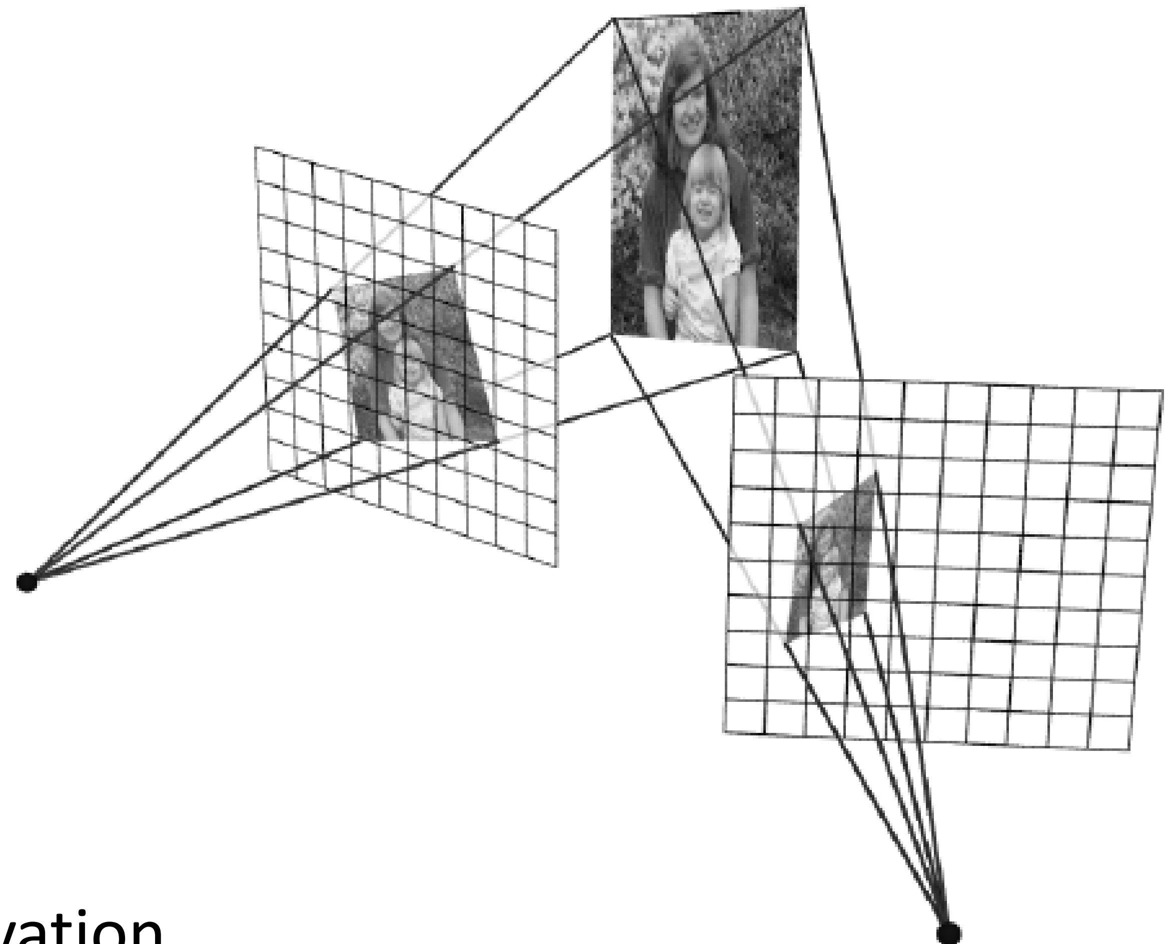


Image reprojection



- Observation
  - Rather than thinking of this as a 3D reprojection, think of it as a 2D image warp from one image to another

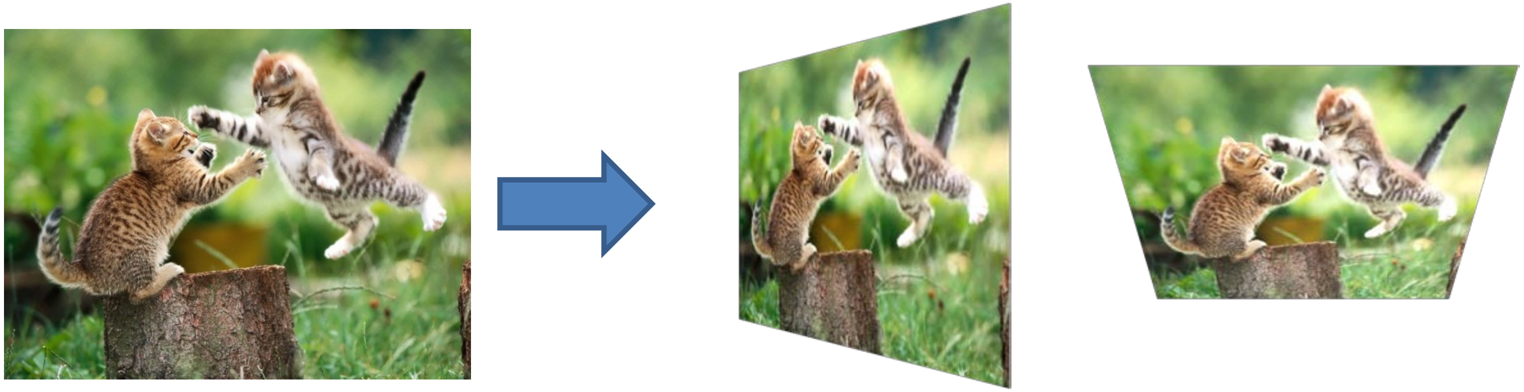
Motion models

- What happens when we take two images with a camera and try to align them?
- translation?
- rotation?
- scale?
- affine?
- Perspective?

Recall: Projective transformations

- (aka *homographies*)

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



Parametric (global) warping

- Examples of parametric warps:

translation

rotation

aspect

affine

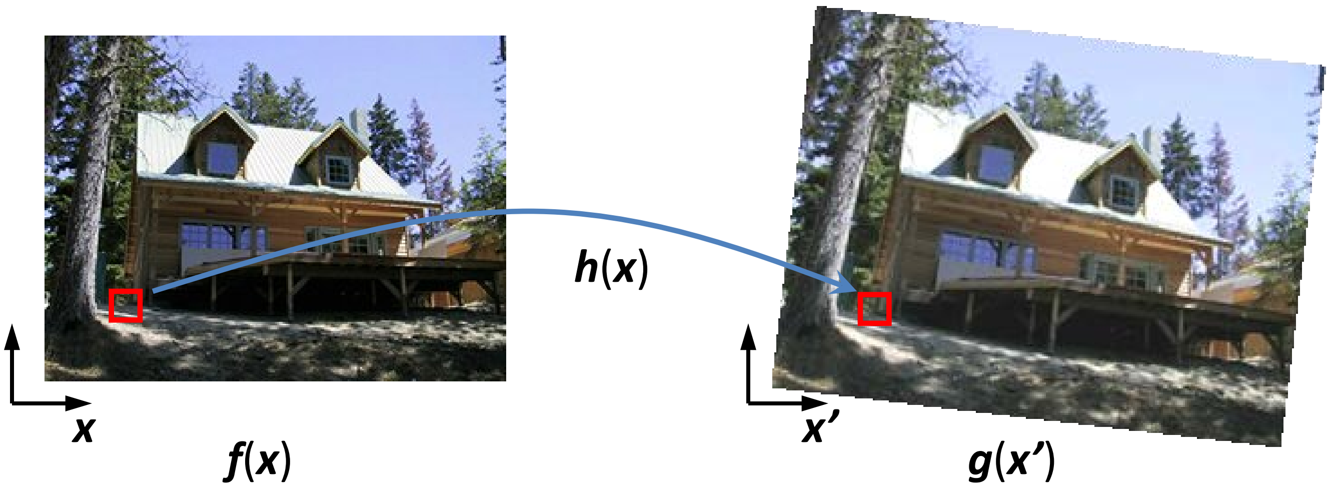
perspective

2D coordinate transformations

- translation: $\mathbf{x}' = \mathbf{x} + \mathbf{t}$ $\qquad \mathbf{x} = (x,y)$
- rotation: $\mathbf{x}' = \mathbf{R}\,\mathbf{x} + \mathbf{t}$
- similarity: $\mathbf{x}' = s\,\mathbf{R}\,\mathbf{x} + \mathbf{t}$
- affine: $\mathbf{x}' = \mathbf{A}\,\mathbf{x} + \mathbf{t}$
- perspective: $\underline{\mathbf{x}}' \cong \mathbf{H}\,\underline{\mathbf{x}}$ $\qquad \underline{\mathbf{x}} = (x,y,1)$
  ($\underline{\mathbf{x}}$ is a *homogeneous* coordinate)

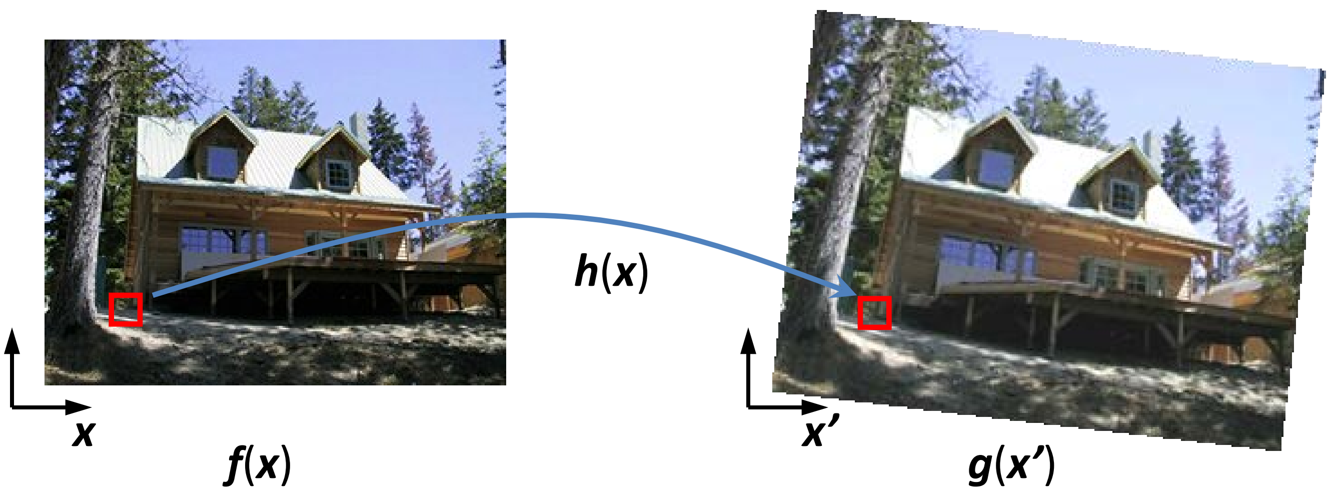
Image Warping

- Given a coordinate transform ***x'*** = ***h***(***x***) and a source image ***f***(***x***), how do we compute a transformed image ***g***(***x'***) = ***f***(***h***(***x***))?



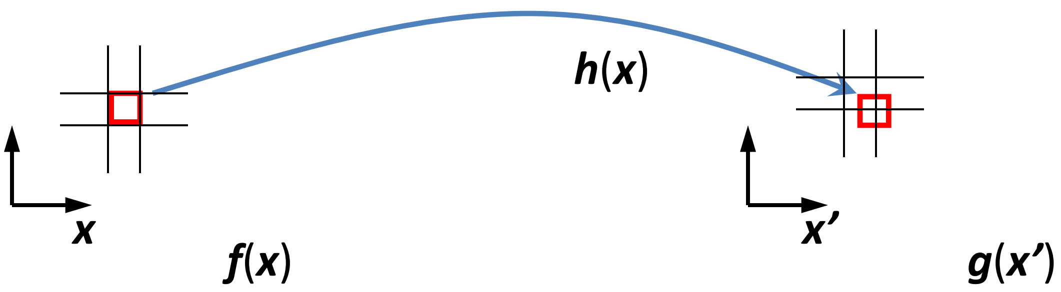
Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x}')$
  - What if pixel lands "between" two pixels?



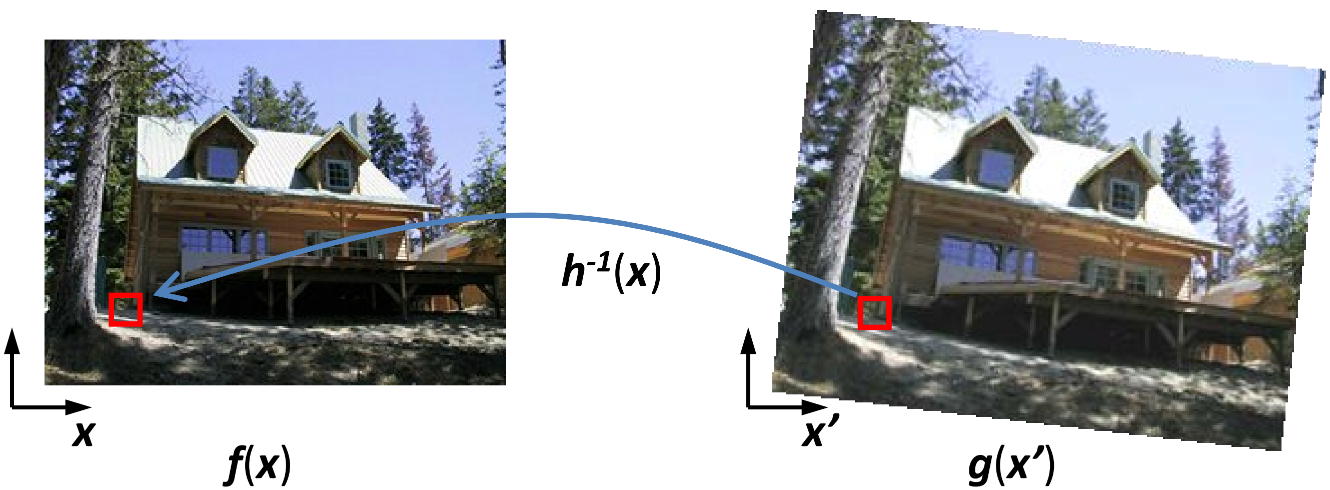
Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x'})$
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)



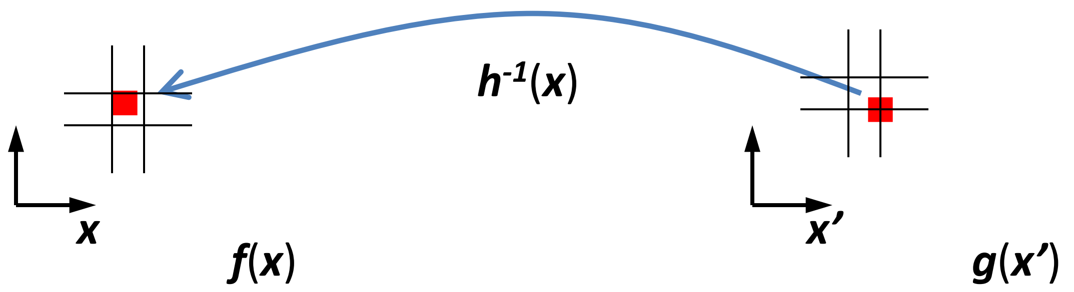
Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ from its corresponding location $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{f}(\boldsymbol{x})$
  - What if pixel comes from “between” two pixels?



Inverse Warping

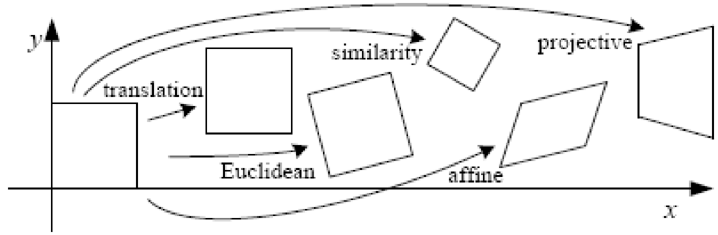
- Get each pixel ***g***(***x'***) from its corresponding location ***x'*** = ***h***(***x***) in ***f***(***x***)
  - What if pixel comes from “between” two pixels?
  - Answer: *resample* color value from *interpolated* source image



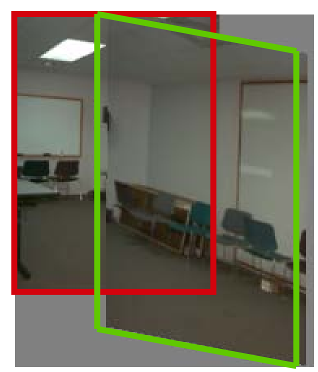
Interpolation

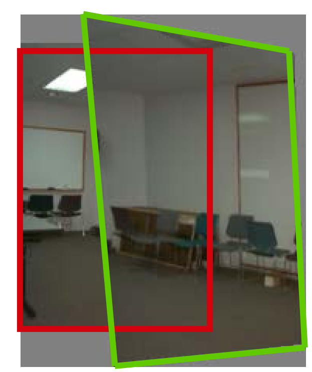
- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)

Motion models



Translation **Affine** **Perspective**





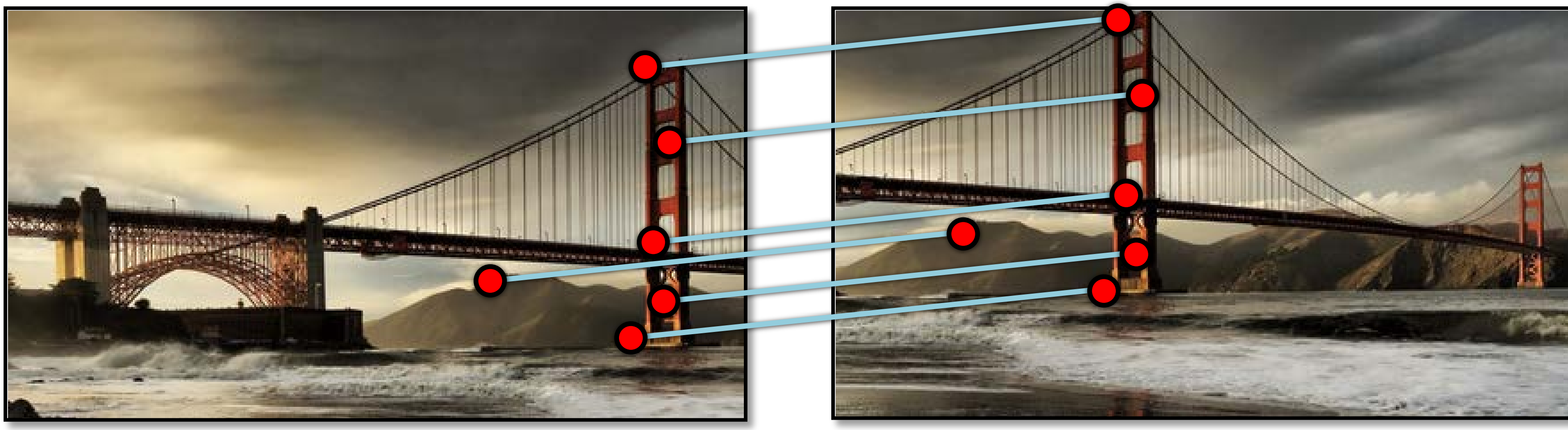
2 unknowns **6 unknowns** **8 unknowns**

Finding the transformation

- Translation = 2 degrees of freedom
- Similarity = 4 degrees of freedom
- Affine = 6 degrees of freedom
- Homography = 8 degrees of freedom

- How many corresponding points do we need to solve?

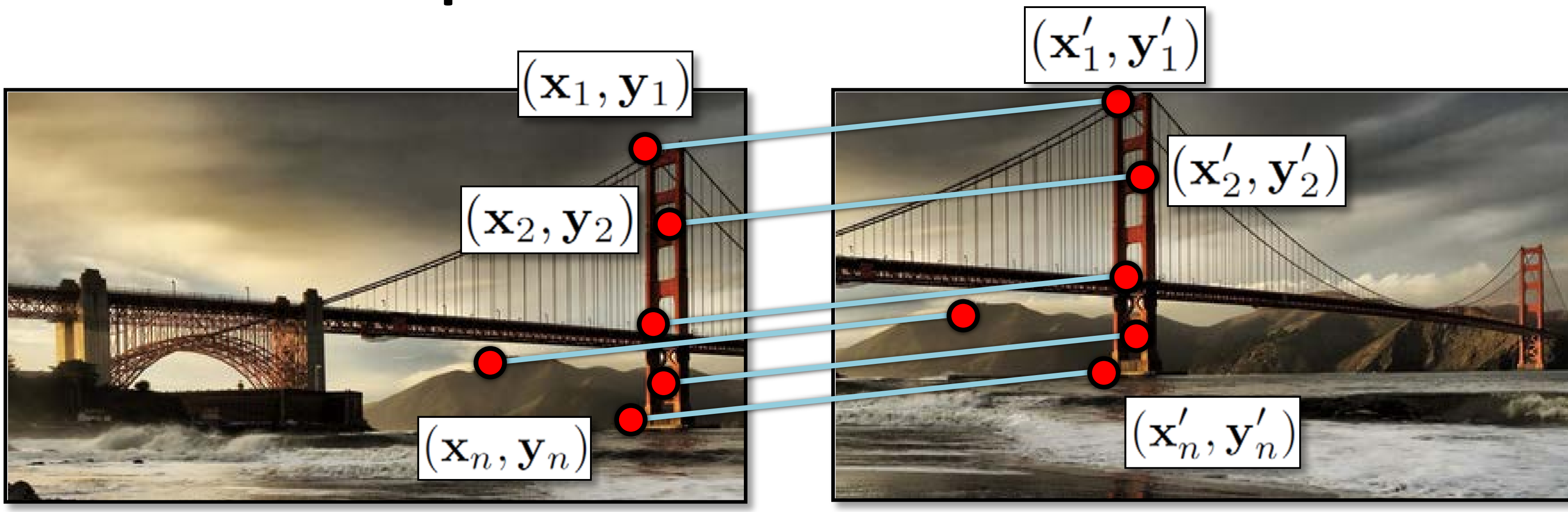
Simple case: translations



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

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

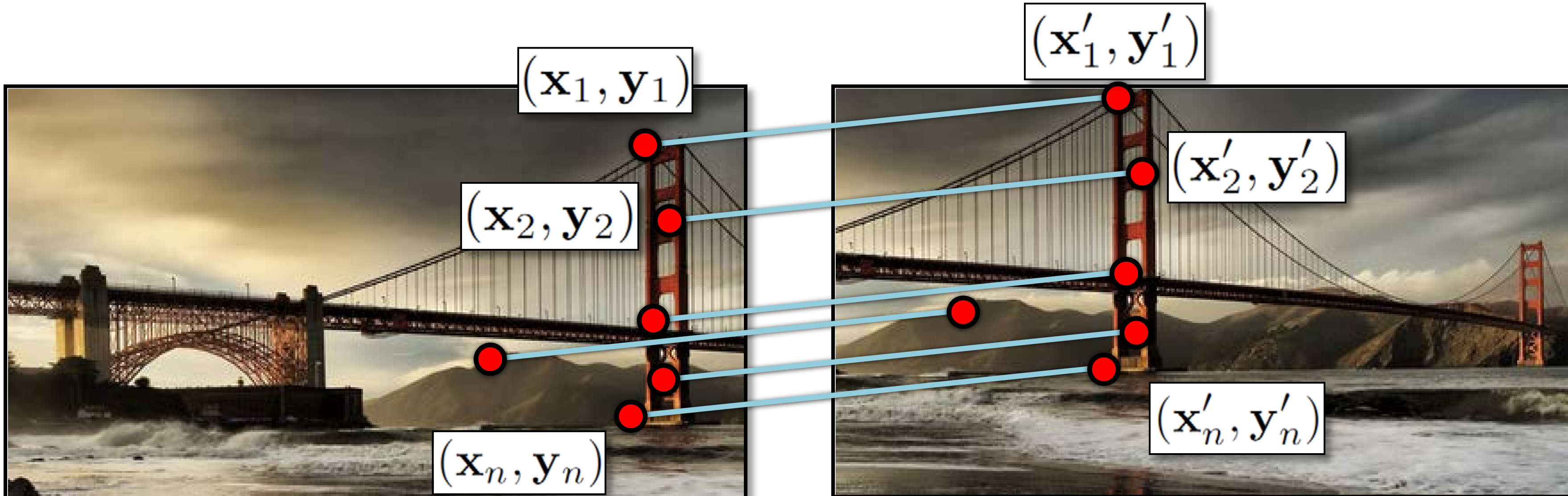
Simple case: translations



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

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

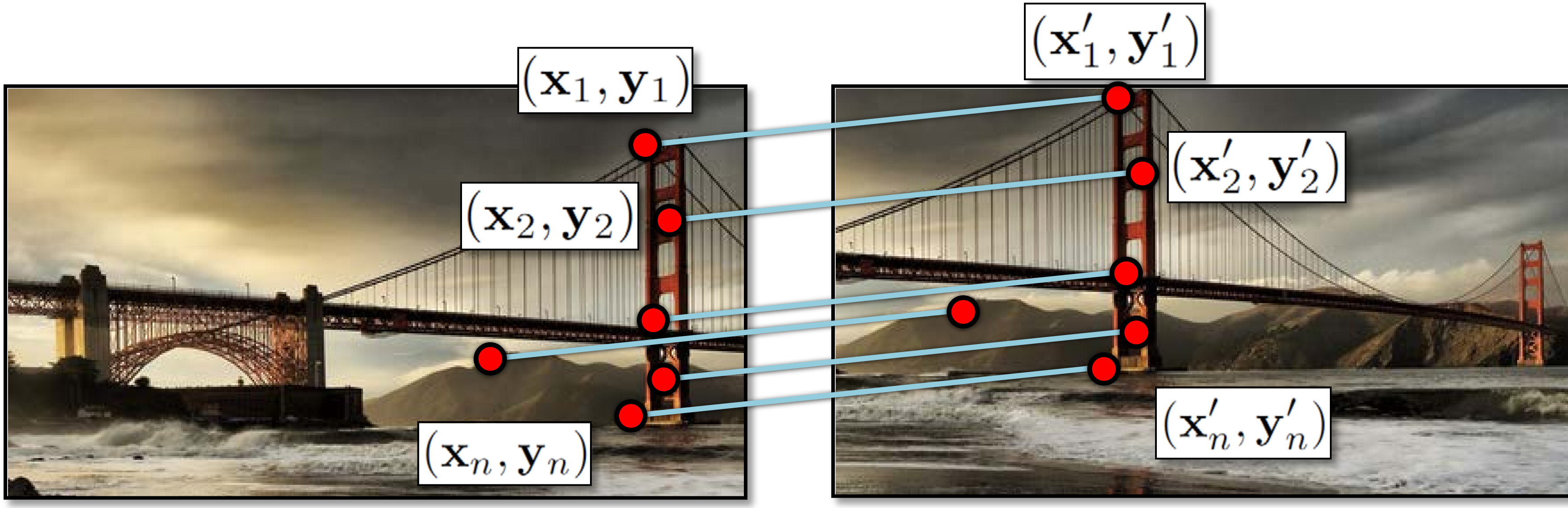
Simple case: translations



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

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

Simple case: translations



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

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

- 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 displacement

Least squares

$$\mathbf{A}\mathbf{t} = \mathbf{b}$$

- Find **t** that minimizes

$$||\mathbf{A}\mathbf{t} - \mathbf{b}||^2$$

- To solve, form the *normal equations*

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

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

Solving for translations

- Using least squares

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

$$\mathbf{A} \quad \mathbf{t} = \mathbf{b}$$

2*n* x 2   2 x 1   2*n* x 1

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

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

$$\mathbf{A} \quad \mathbf{t} = \mathbf{b}$$

Solving for homographies

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

Why is this now a variable and not just 1?

- A homography is a projective object, in that it has no scale.
  It is represented by a 3 x 2 matrix, up to scale.

- One way of fixing the scale is to set one of the coordinates to 1, though that choice is arbitrary.

- But that's what most people do and your assignment code does.

Solving for homographies

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

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

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

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

Solving for homographies

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

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

Direct Linear Transforms

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

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

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

- 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

Direct Linear Transforms

- Why could we not solve for the homography in exactly the same way we did for the affine transform, ie.

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

Answer from Sameer

- For an affine transform, we have equations of the form $Ax_i + b = y_i$, solvable by linear regression.
- For the homography, the equation is of the form

  $H\tilde{x}_i \sim \tilde{y}_i$ (homogeneous coordinates)

and the ~ means it holds only up to scale. The affine solution does not hold.

- To get rid of the scale ambiguity, we can break up the H into 3 row vectors and divide out the third coordinate to make it 1.

  $[h_1' \tilde{x}_i / h_3' \tilde{x}_i , h_2' \tilde{x}_i / h_3' \tilde{x}_i] = [y_{i1}, y_{i2}]$ for each i.

Continued

- Expanding these out leads to (for each i)

  $h_1' \tilde{x}_i - y_{i1} h_3' \tilde{x}_i = 0$

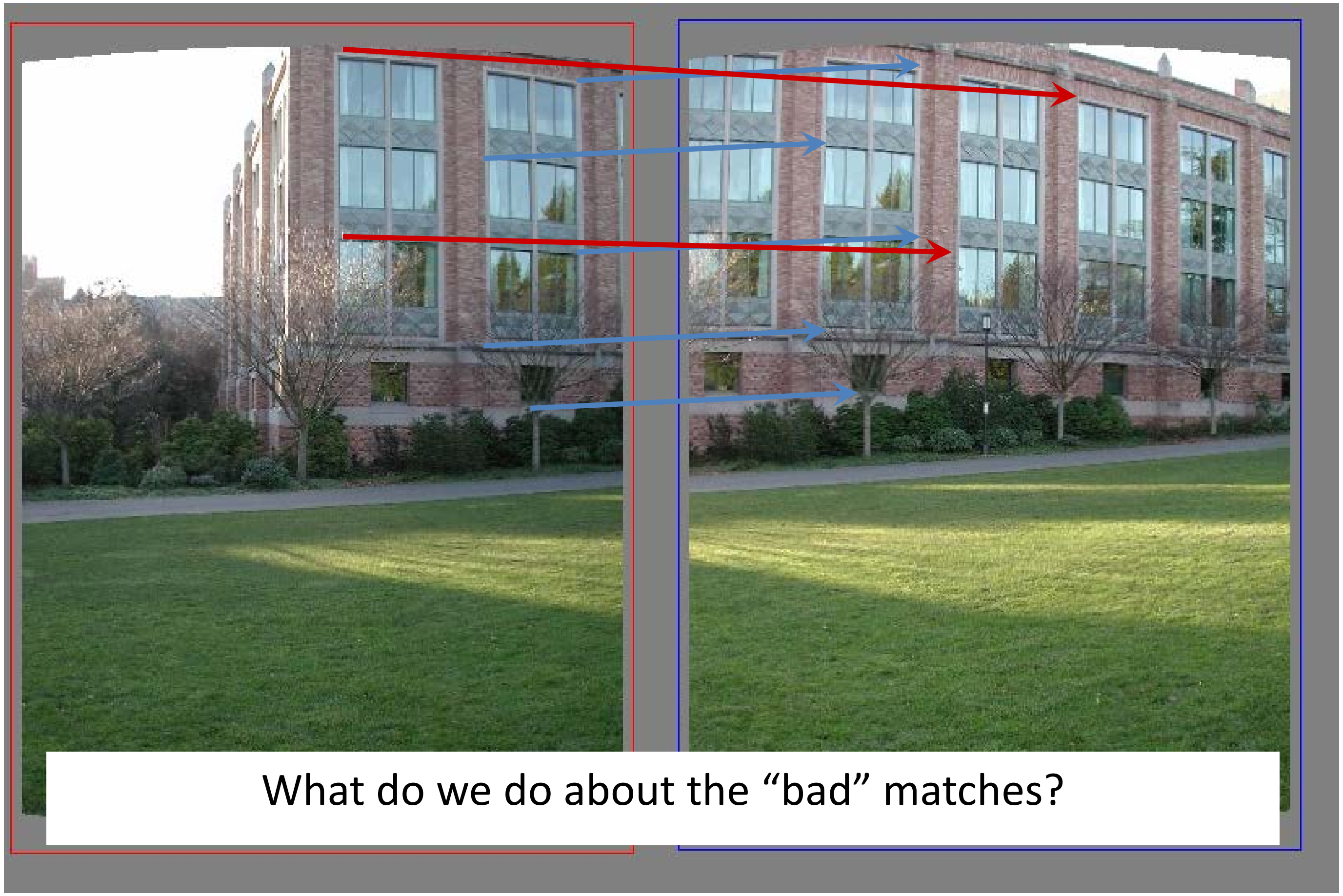
  $h_2' \tilde{x}_i - y_{i2} h_3' \tilde{x}_i = 0$

a system with no constant terms.

- The resultant system to be solved is of the form A h = 0

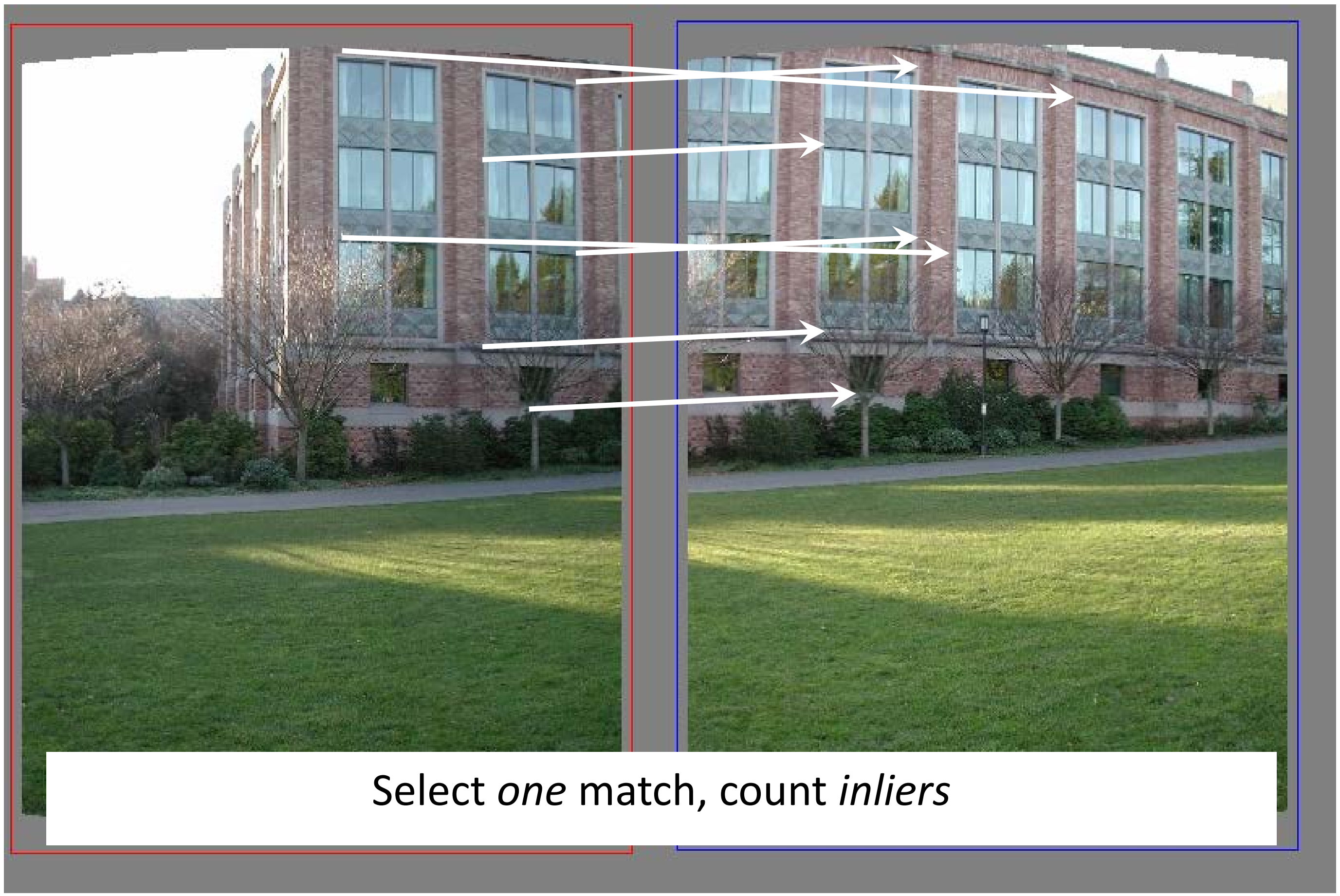
- Then this is solved with the eigenvector approach.

Matching features



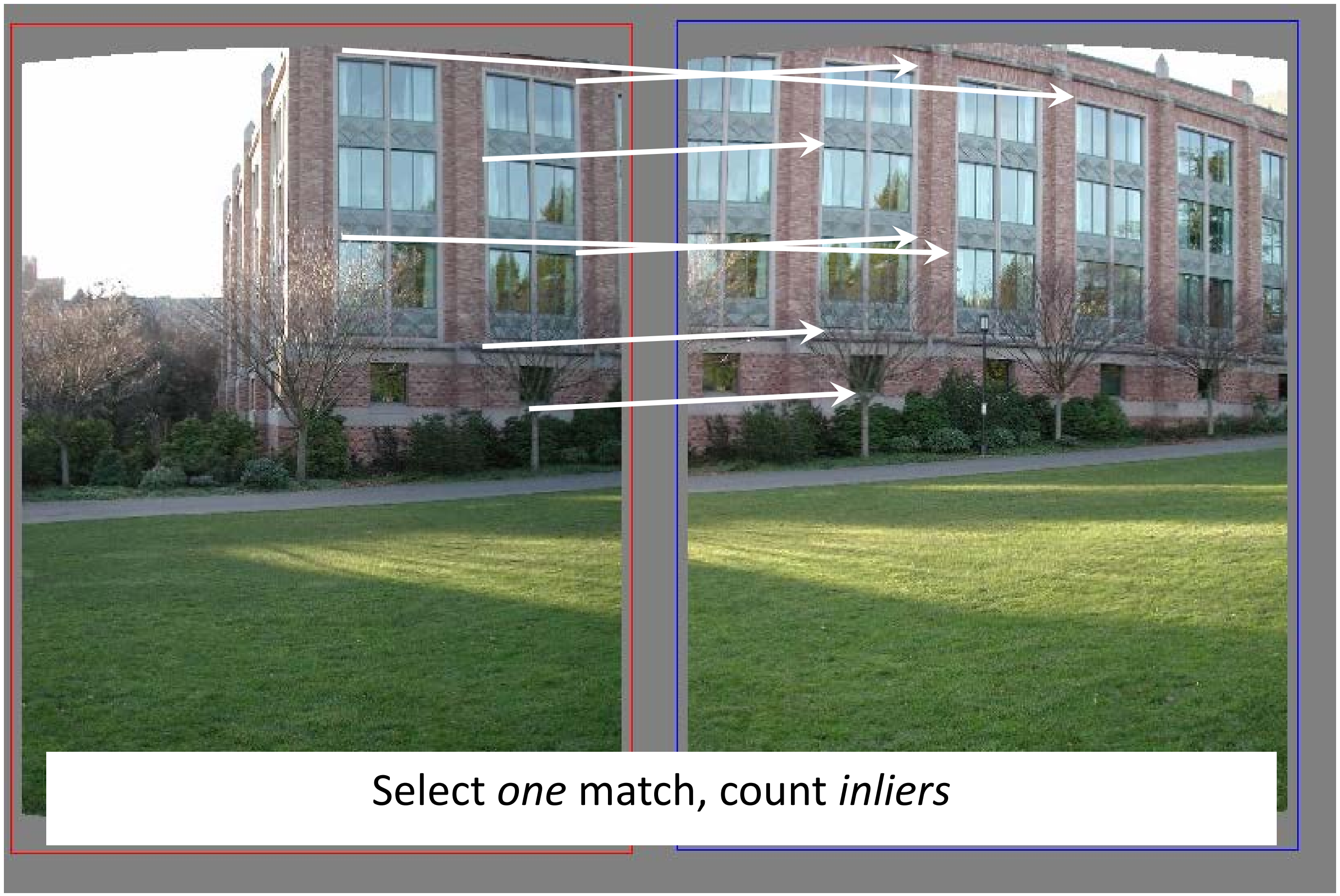
What do we do about the “bad” matches?

RAndom SAmple Consensus

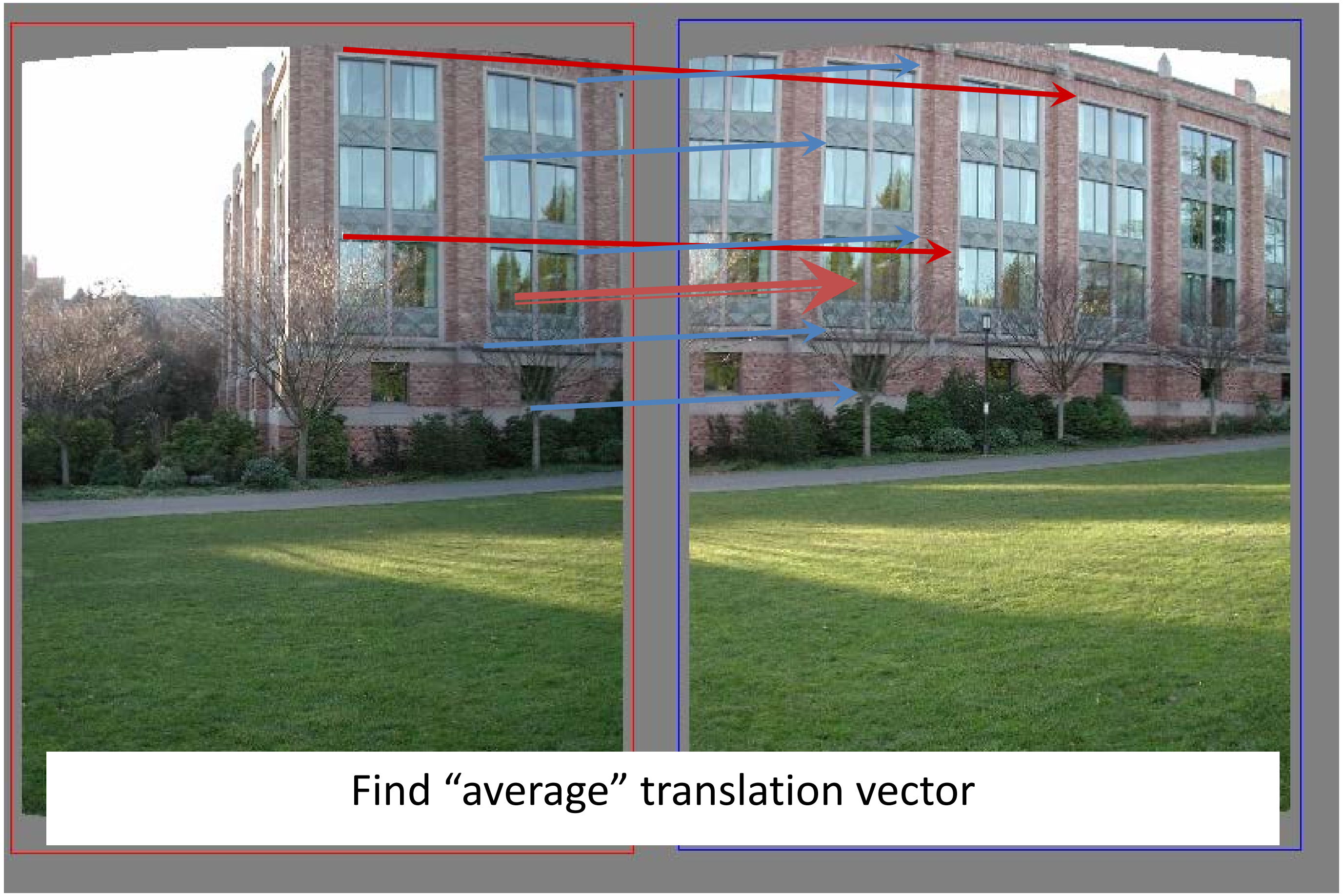


Select *one* match, count *inliers*

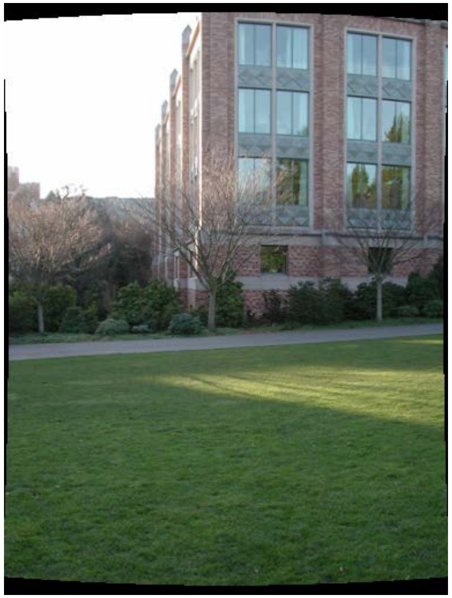
RAndom SAmple Consensus

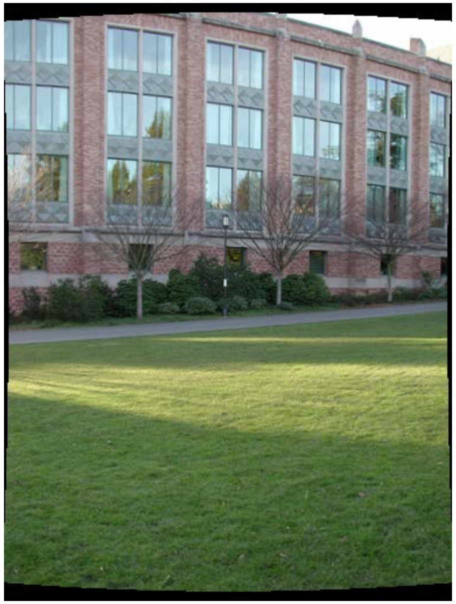


Least squares fit



Find “average” translation vector





RANSAC for estimating homography

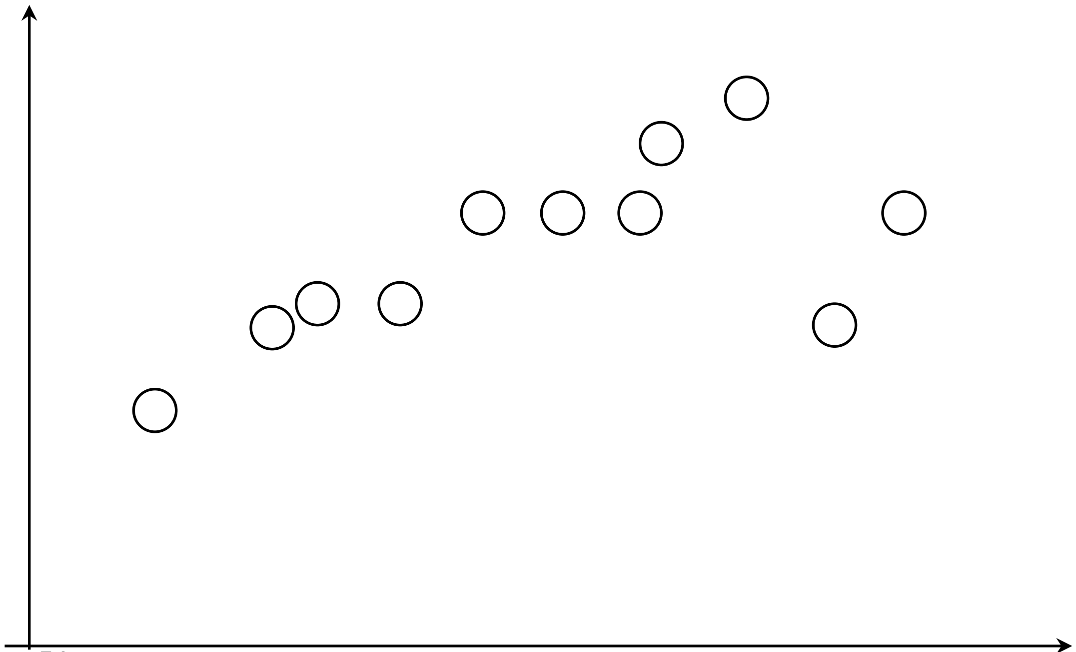
- RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography $\boldsymbol{H}$ (exact)
3. Compute inliers where $\|p_i', \boldsymbol{H}\, p_i\| < \varepsilon$

- Keep largest set of inliers
- Re-compute least-squares $\boldsymbol{H}$ estimate using all of the inliers

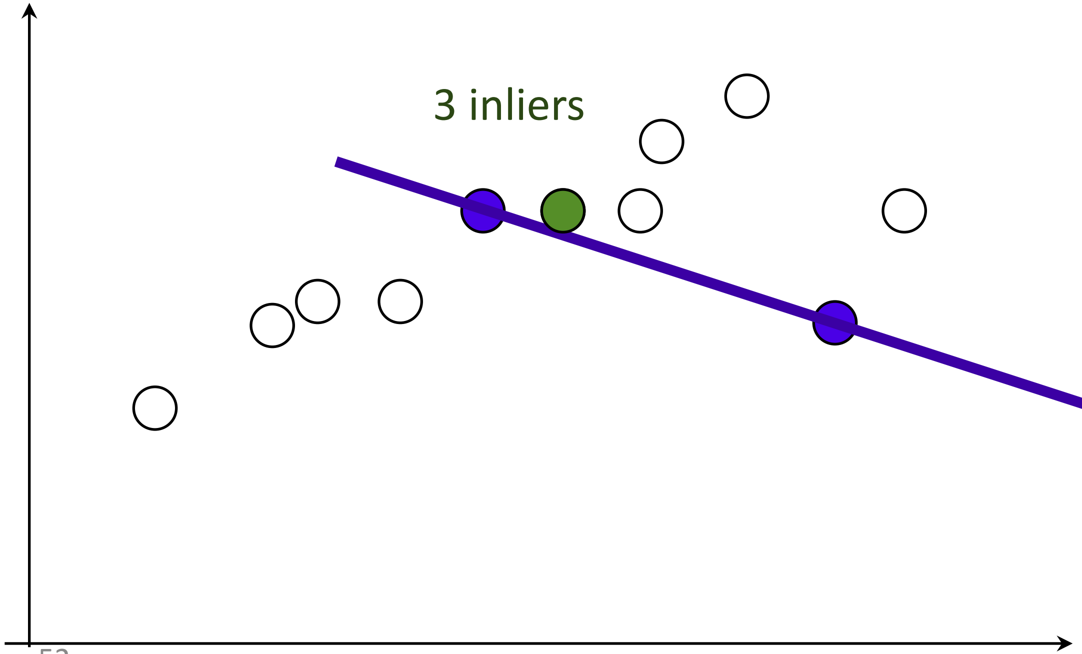
Simple example: fit a line

- Rather than homography H (8 numbers) fit y=ax+b (2 numbers a, b) to 2D pairs



Simple example: fit a line

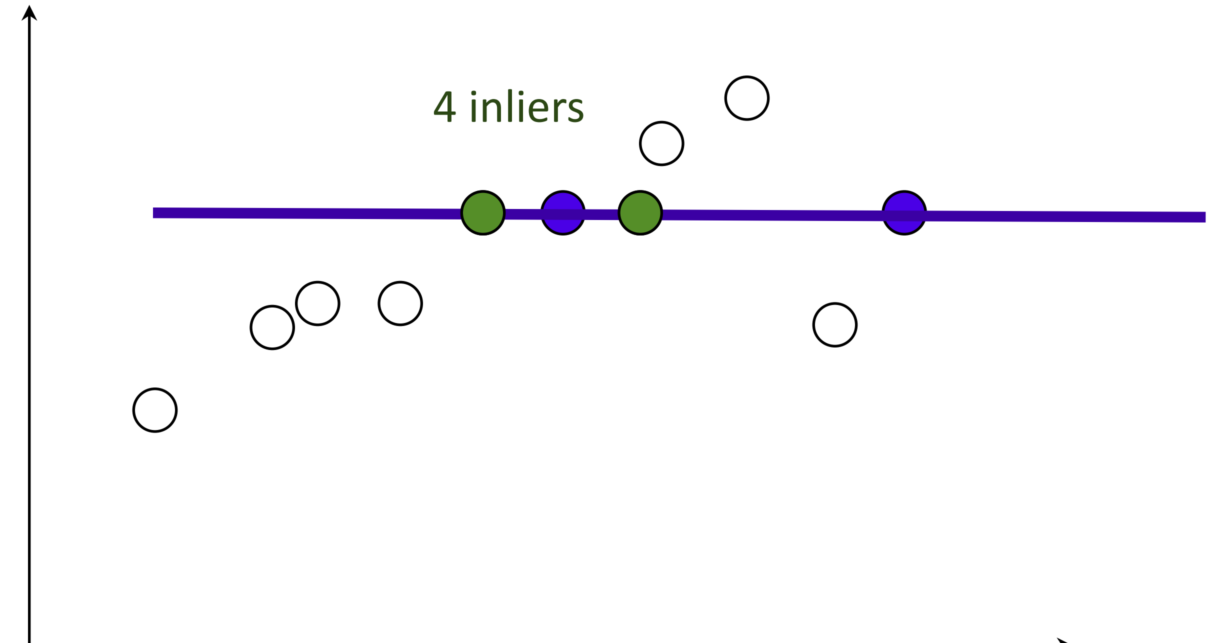
- Pick 2 points
- Fit line
- Count inliers



52

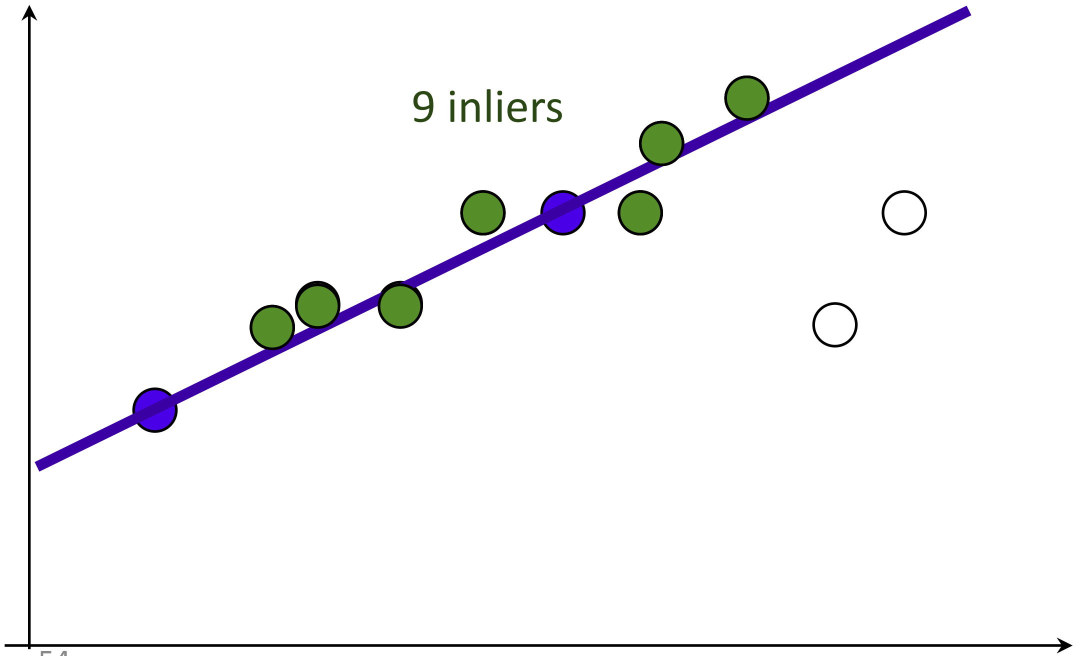
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



Simple example: fit a line

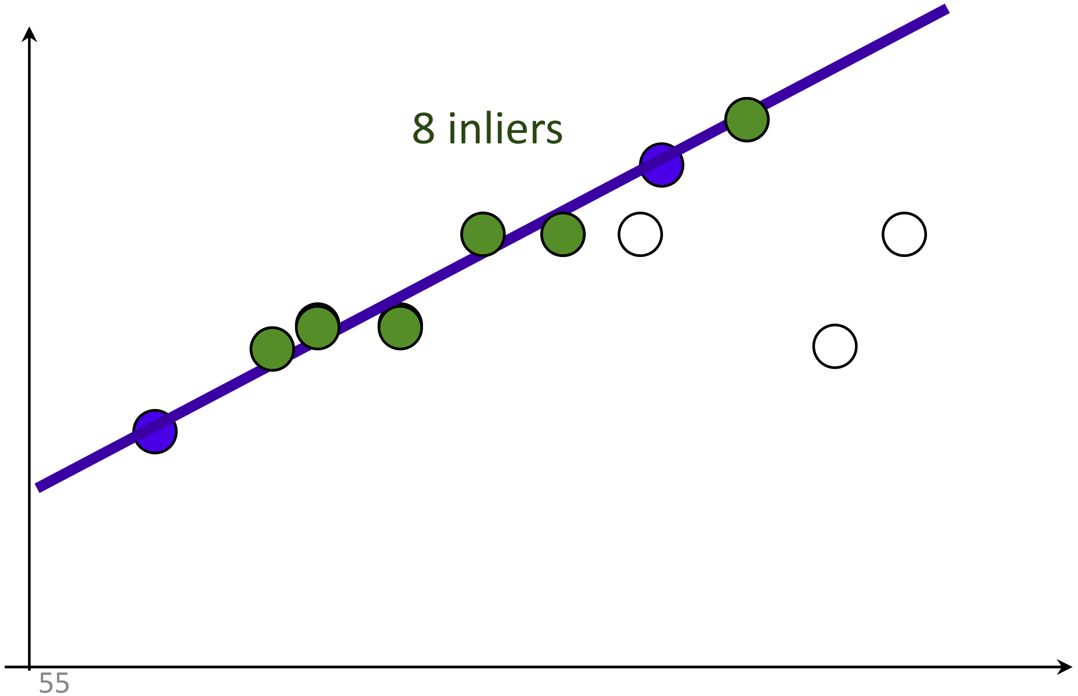
- Pick 2 points
- Fit line
- Count inliers



54

Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



Simple example: fit a line

- Use biggest set of inliers
- Do least-square fit