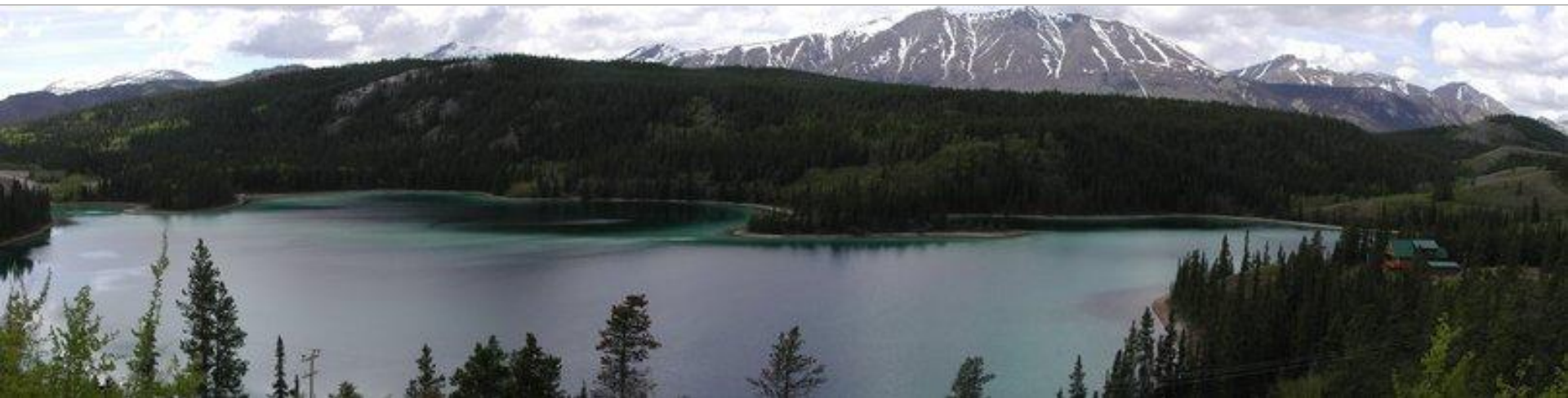
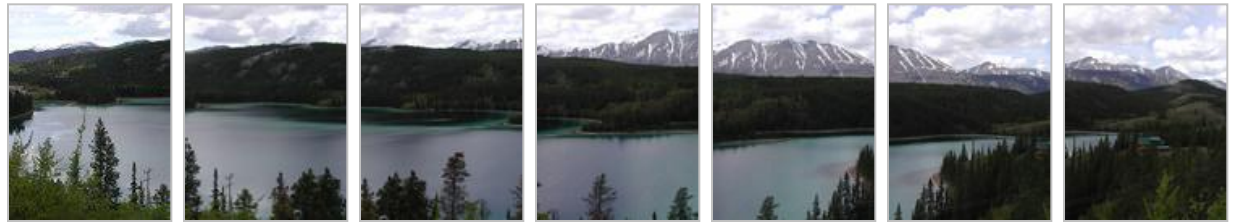


Image Stitching

Ali Farhadi
CSE 455

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



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

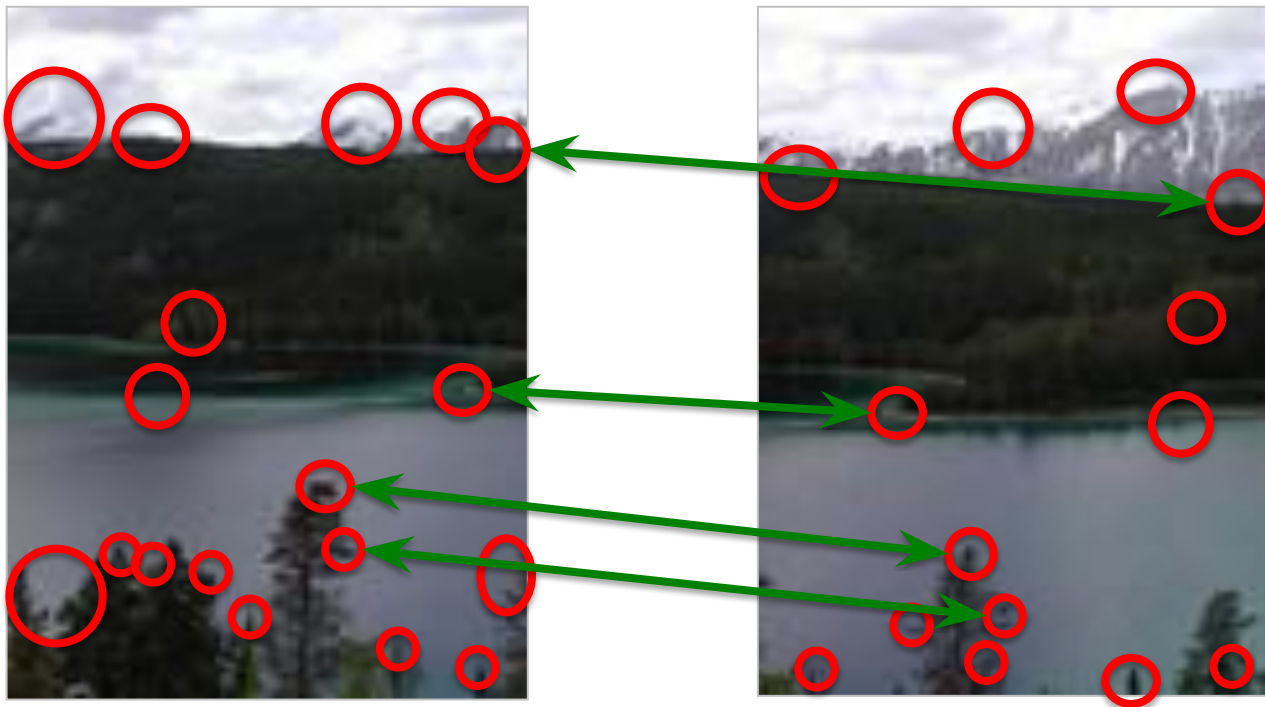
1. Take a sequence of images from the same position

- 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

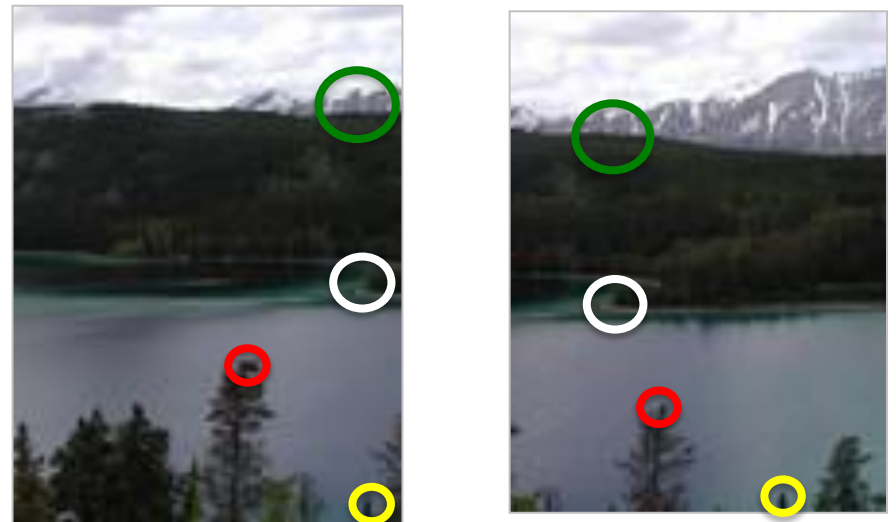
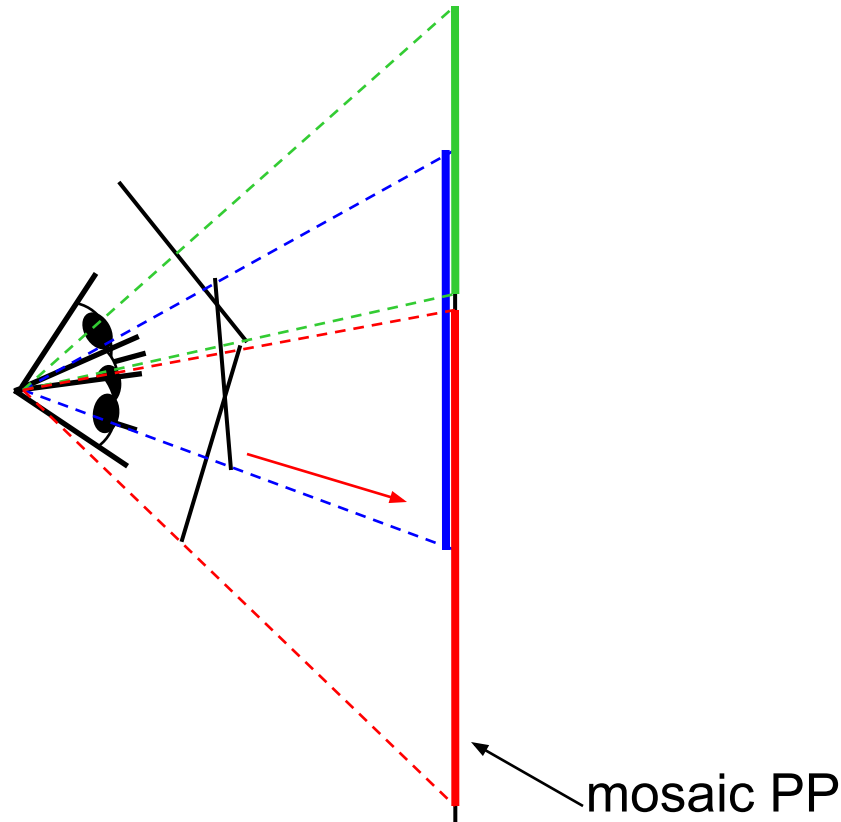


Image reprojection



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

Example

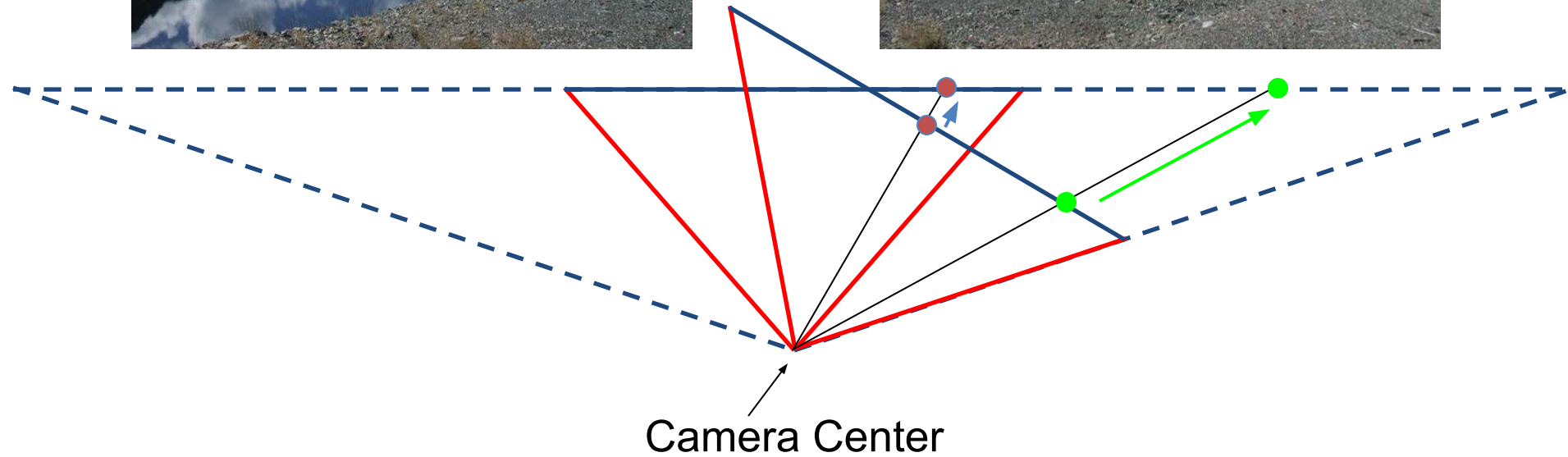
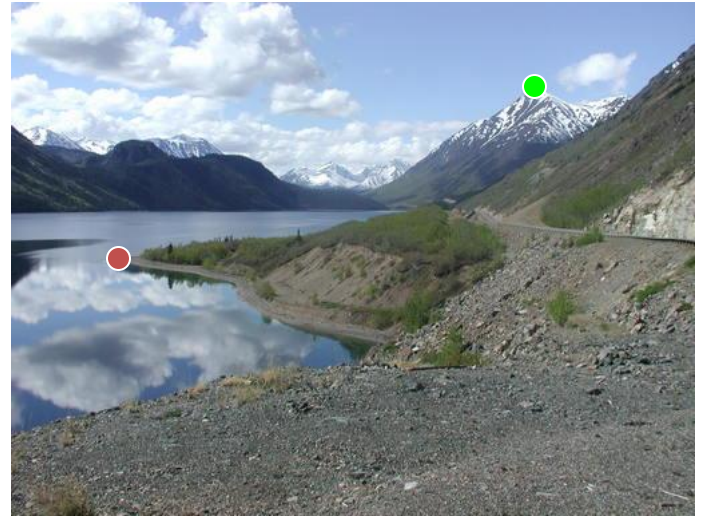
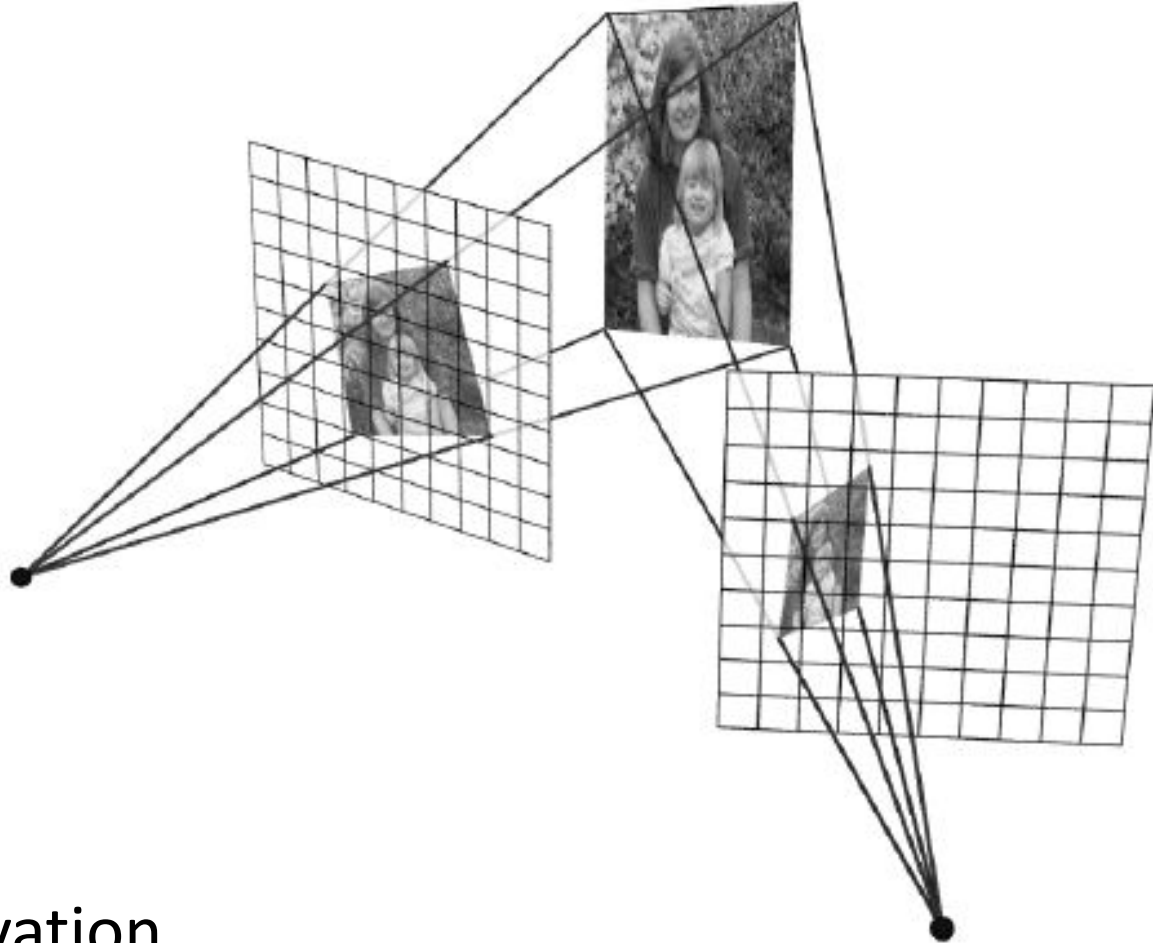


Image reprojection



- 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*)

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



Parametric (global) warping

- Examples of parametric warps:



translation



rotation



aspect



affine



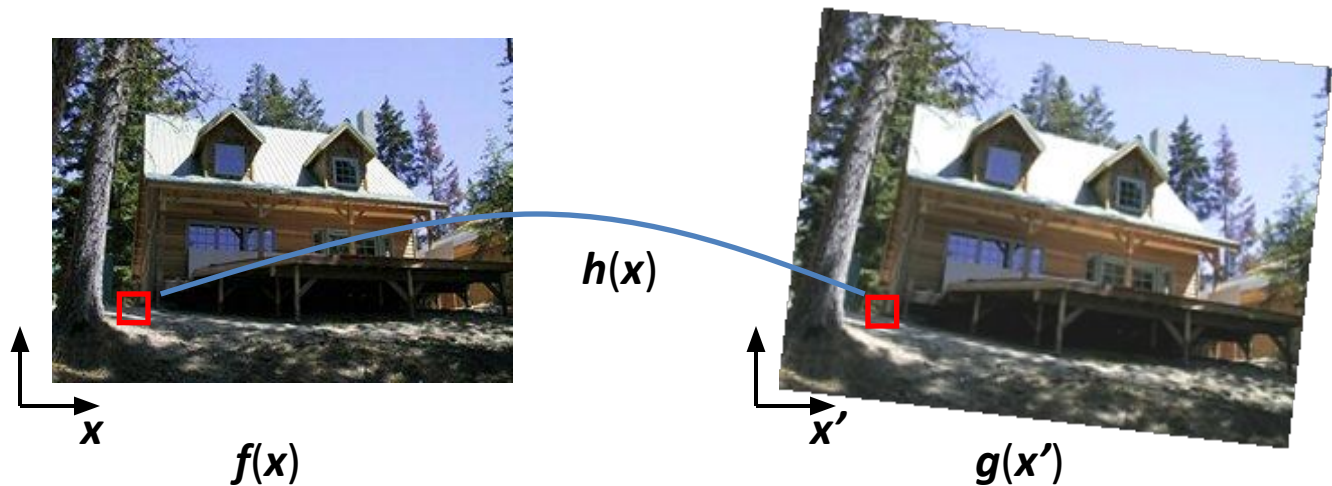
perspective

2D coordinate transformations

- translation: $\mathbf{x}' = \mathbf{x} + \mathbf{t}$ $\mathbf{x} = (x, y)$
- 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: $\underline{\mathbf{x}}' \cong \mathbf{H} \underline{\mathbf{x}}$ $\underline{\mathbf{x}} = (x, y, 1)$
($\underline{\mathbf{x}}$ is a *homogeneous* coordinate)

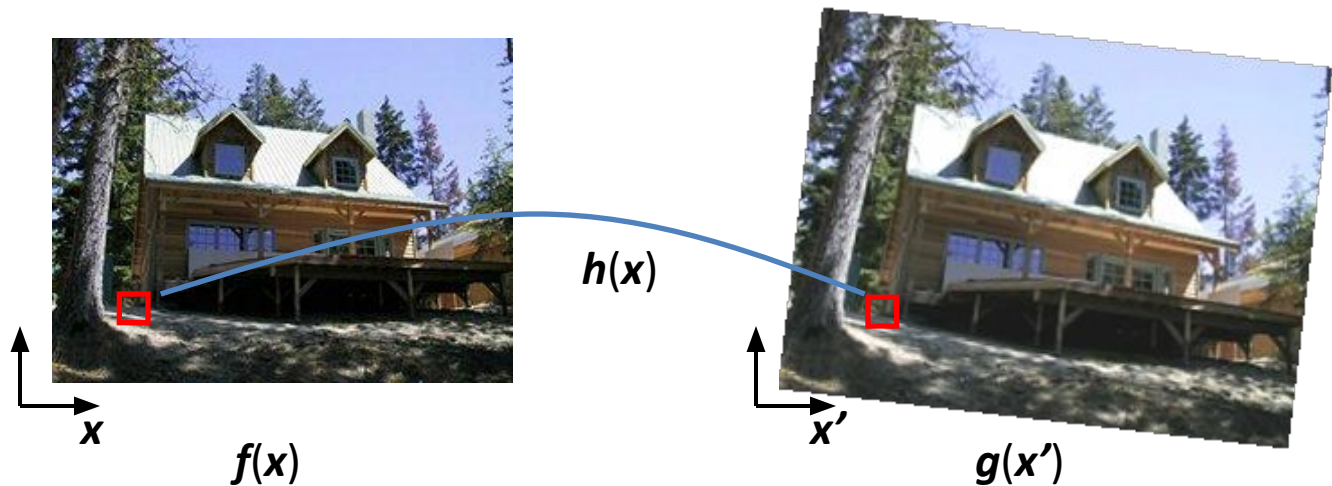
Image Warping

- Given a coordinate transform $\mathbf{x}' = \mathbf{h}(\mathbf{x})$ and a source image $\mathbf{f}(\mathbf{x})$, how do we compute a transformed image $\mathbf{g}(\mathbf{x}') = \mathbf{f}(\mathbf{h}(\mathbf{x}))$?



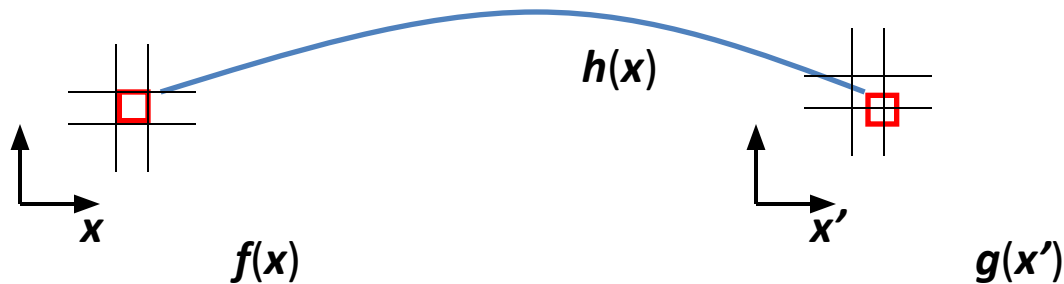
Forward Warping

- Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
- What if pixel lands “between” two pixels?



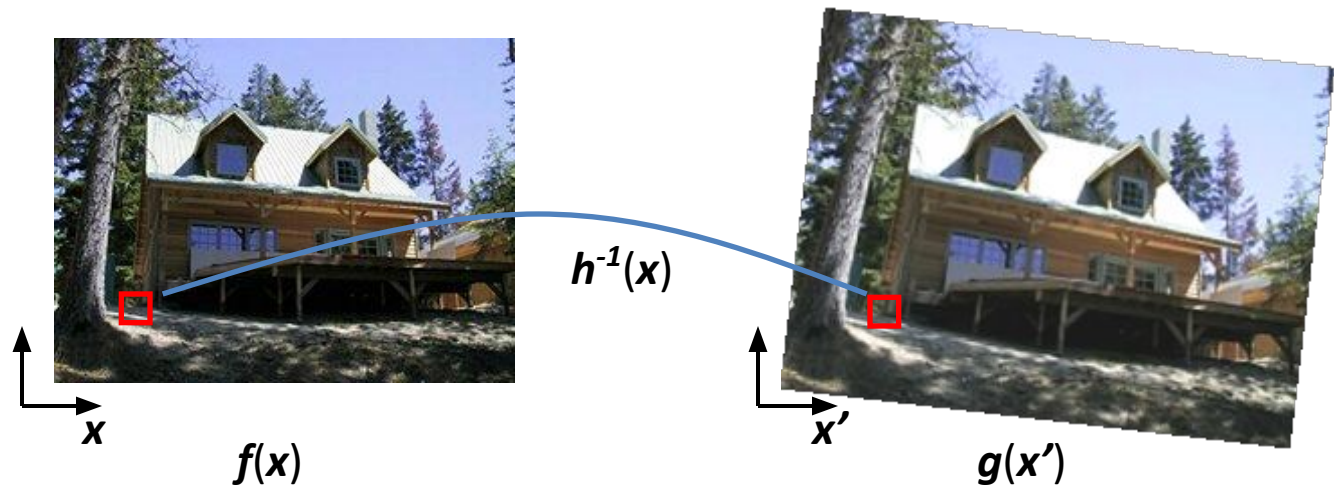
Forward Warping

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



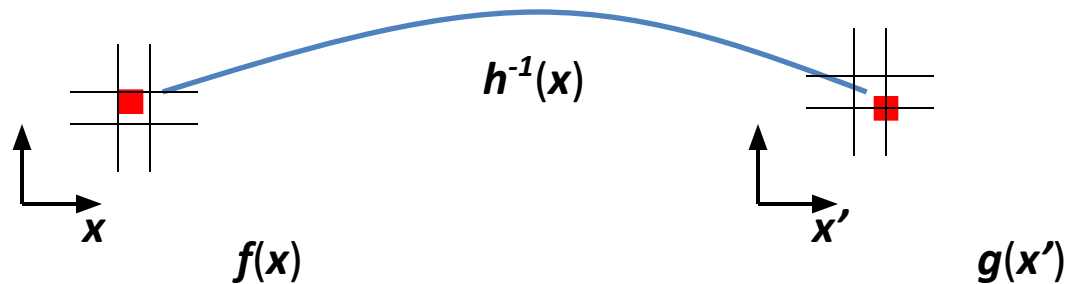
Inverse Warping

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



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

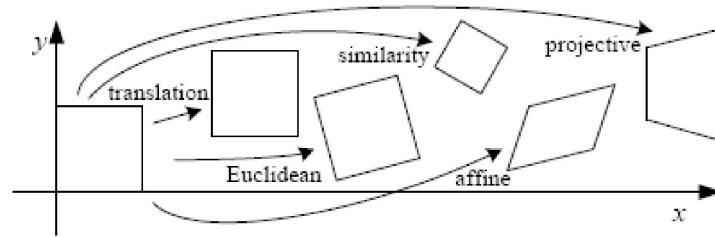


Interpolation

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



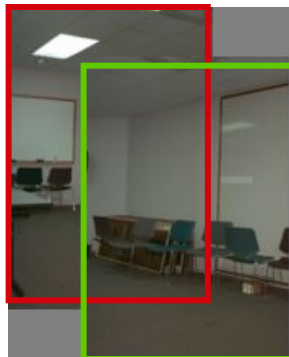
Motion models



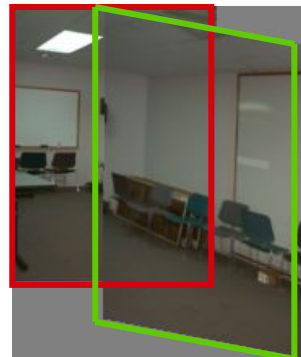
Translation

Affine

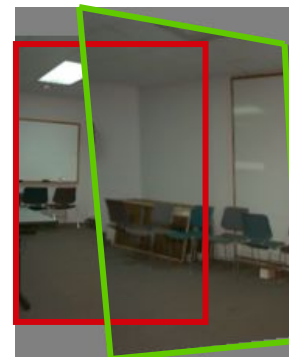
Perspective



2 unknowns



6 unknowns



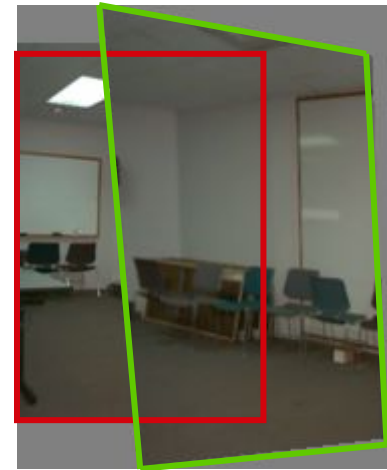
8 unknowns

Finding the transformation

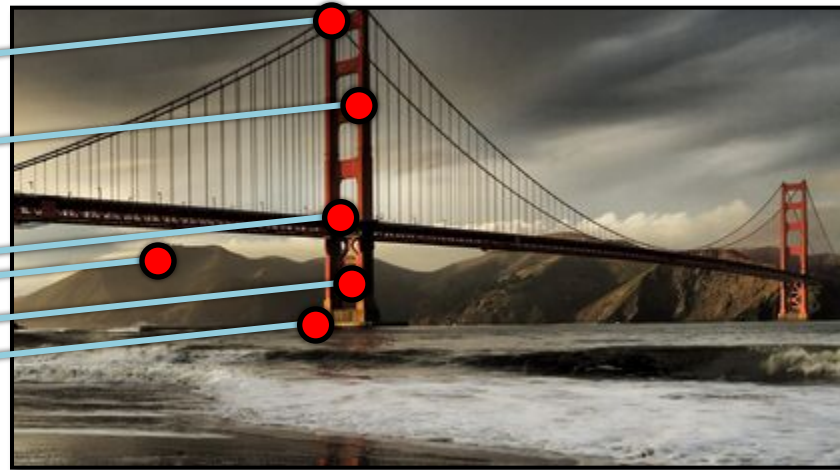
- Translation = 2 degrees of freedom
 - Similarity = 4 degrees of freedom
 - Affine = 6 degrees of freedom
 - Homography = 8 degrees of freedom
-
- How many corresponding points do we need to solve?

Plane perspective mosaics

- 8-parameter generalization of affine motion
 - works for pure rotation or planar surfaces
- **Limitations:**
 - local minima
 - slow convergence
 - difficult to control interactively



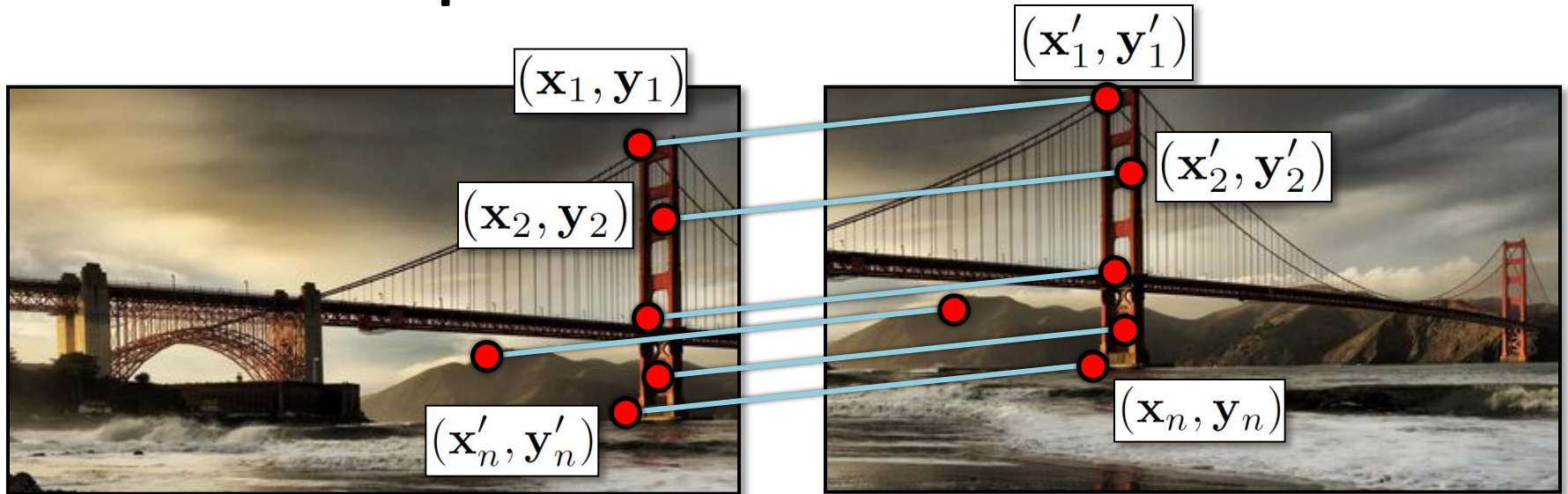
Simple case: translations



(x_t, y_t)

How do we solve for
 (x_t, y_t) ?

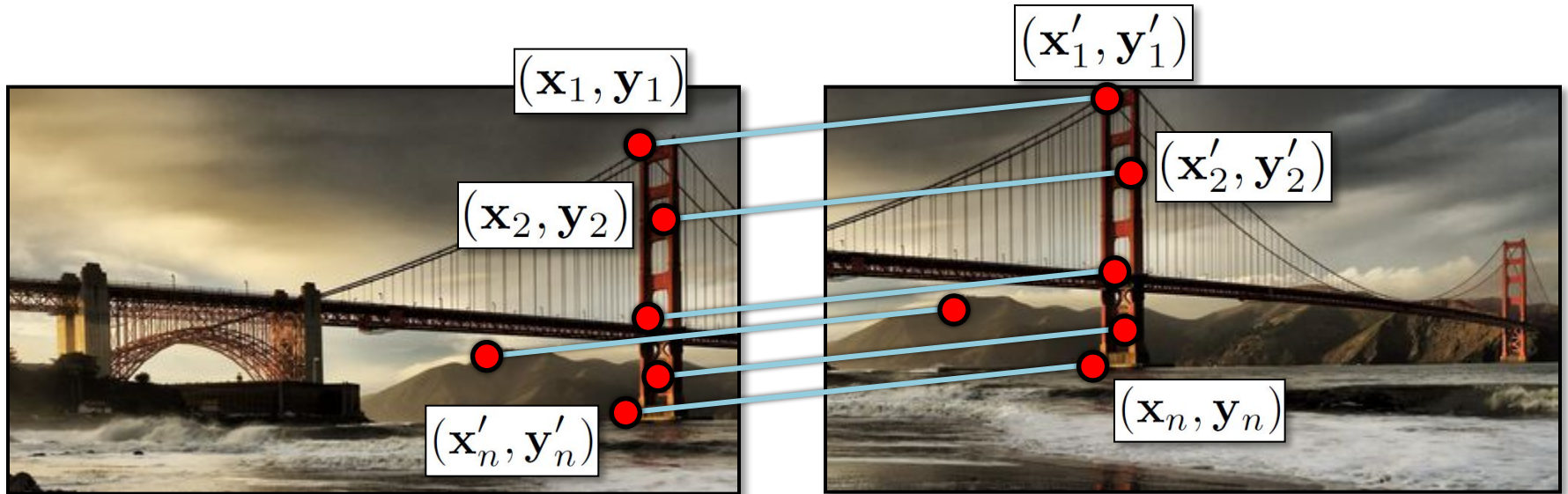
Simple case: translations



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



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

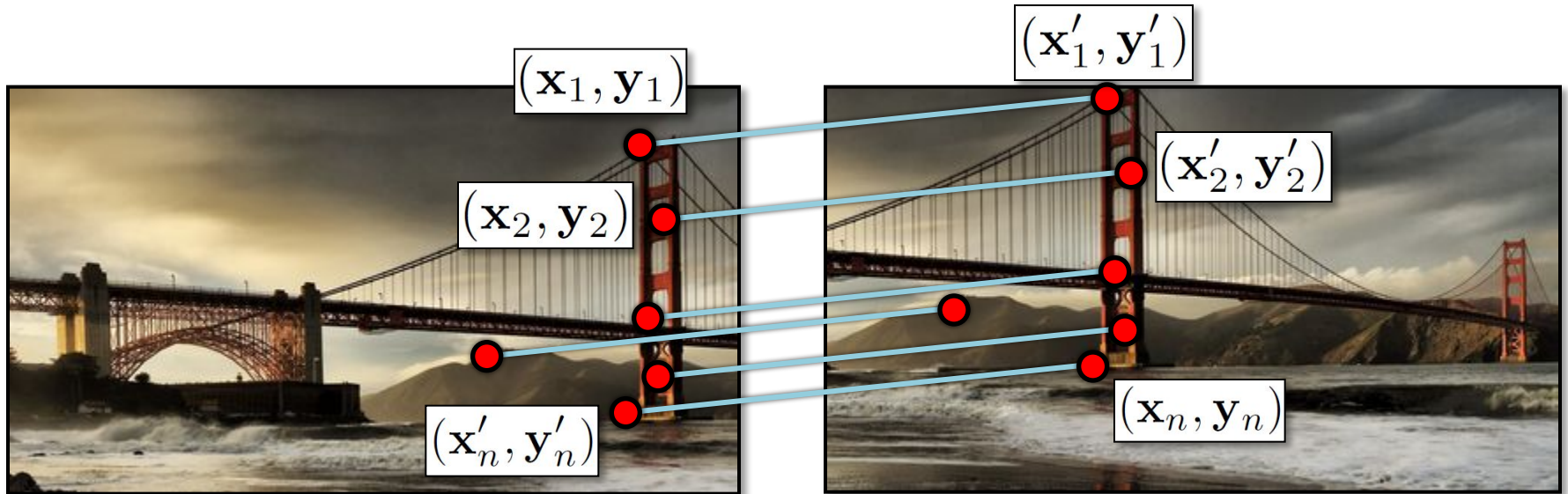
$$\mathbf{y}_i + \mathbf{y}_t = \mathbf{y}'_i$$

- System of linear equations

- What are the knowns? Unknowns?

- How many unknowns? How many equations (per match)?

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

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

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

- we define the *residuals* as

$$r_{\mathbf{x}_i}(\mathbf{x}_t) = (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}'_i$$

$$r_{\mathbf{y}_i}(\mathbf{y}_t) = (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}'_i$$

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 \mathbf{t} that minimizes

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

- To solve, form the *normal equations*

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

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

Solving for translations

- Using least squares

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

$$\mathbf{A}$$

$2n \times 2$

$$\mathbf{t}$$

2×1

$$=$$

$$\mathbf{b}$$

$2n \times 1$

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:

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

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

- Cost function:

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

Affine transformations

- Matrix form

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

A
 $2n \times$

t
 $6 \times$

b
 $2n$

Solving for homographies

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

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

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

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

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

Solving for homographies

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

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

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

Direct Linear Transforms

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

A

$2n \times 9$

h

9

0

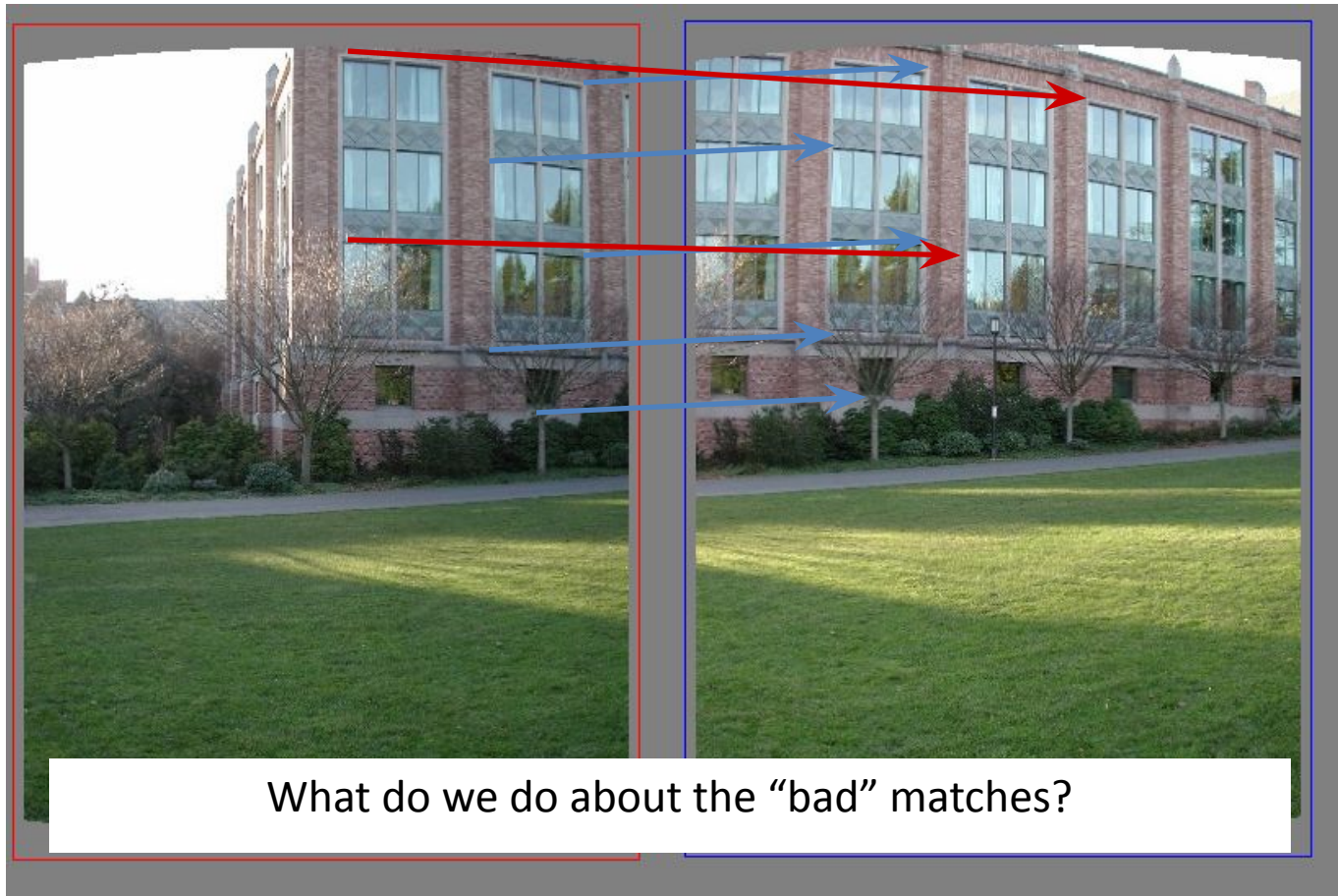
$2n$

Defines a least squares problem:

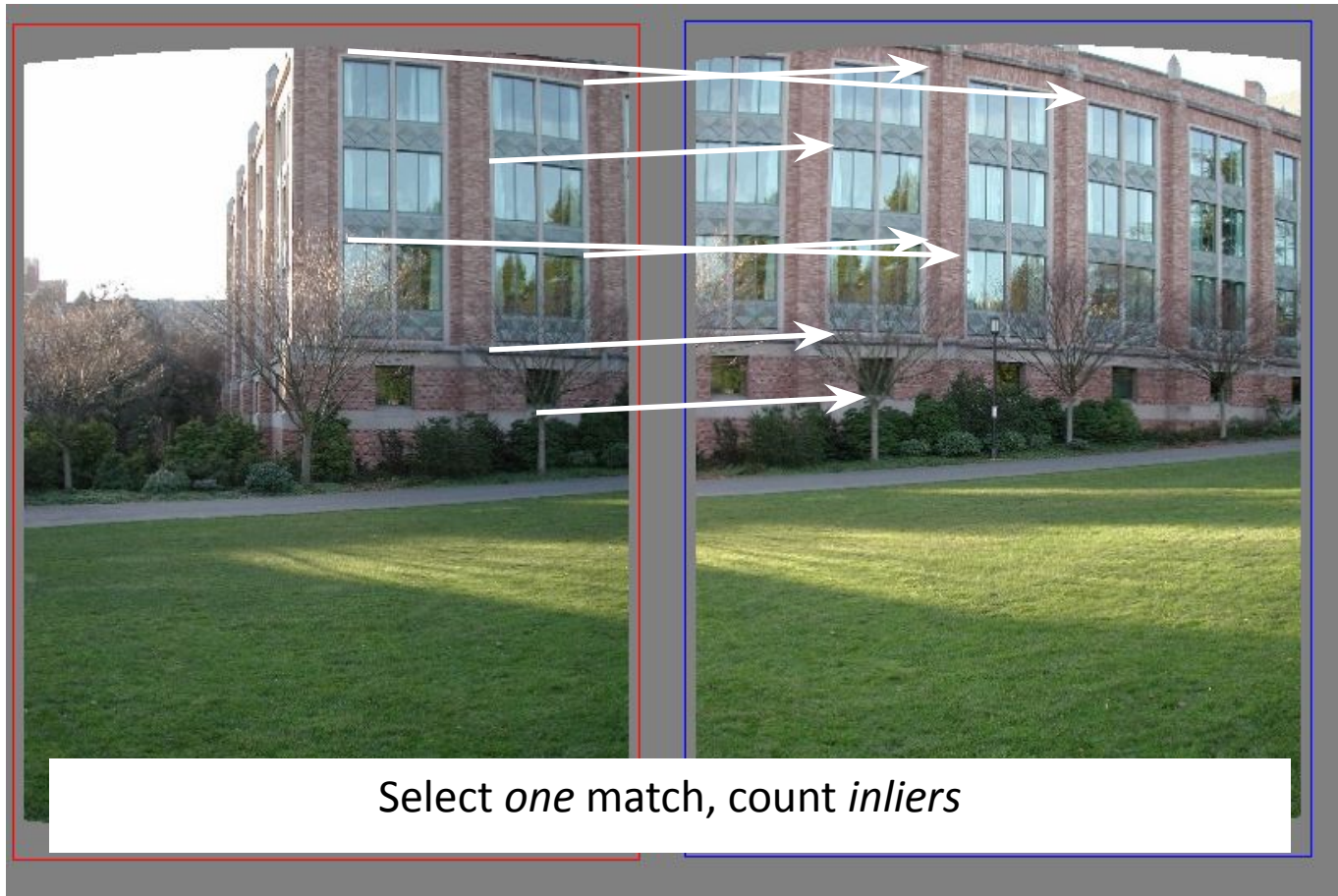
$$\text{minimize } \|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$$

- 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

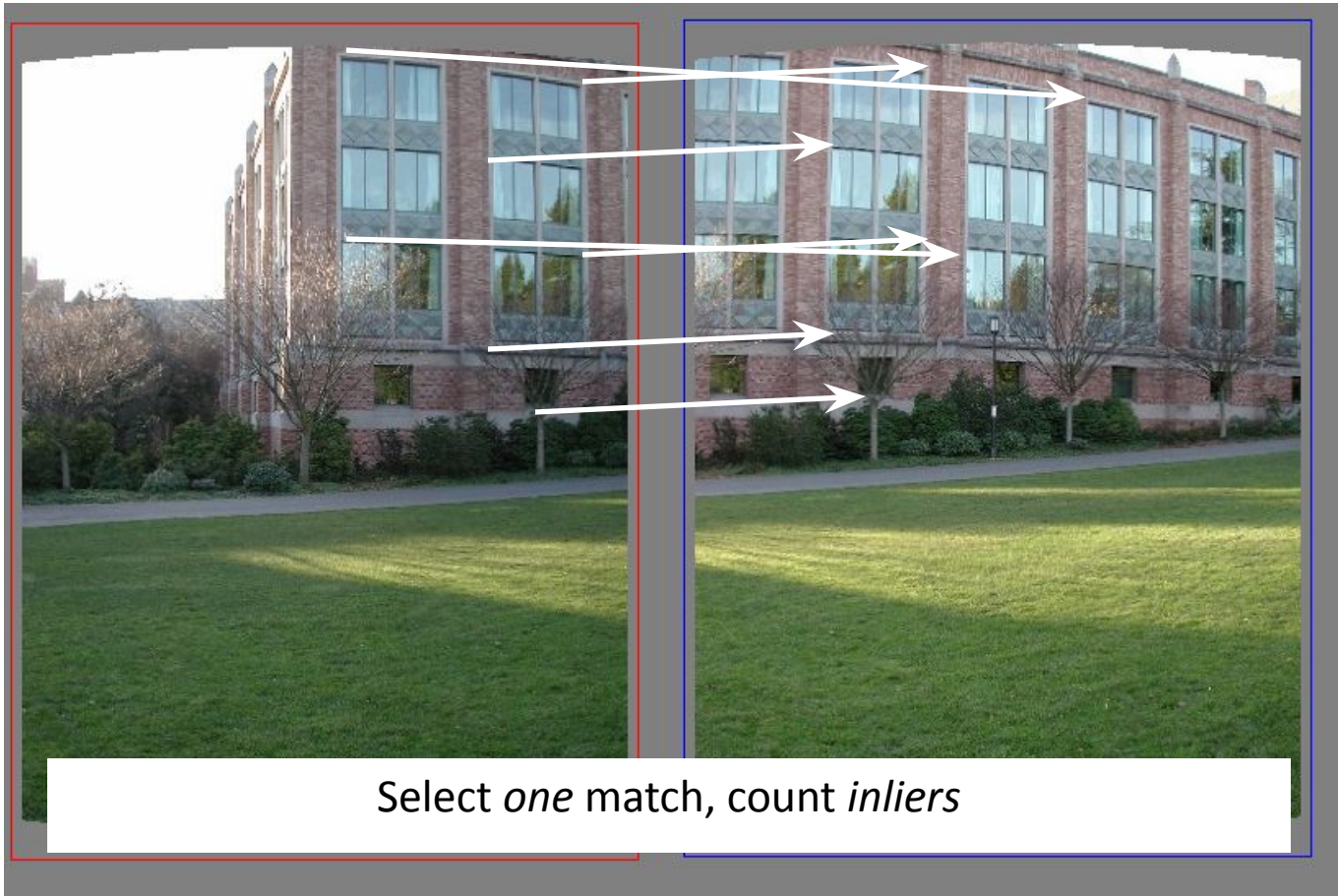
Matching features



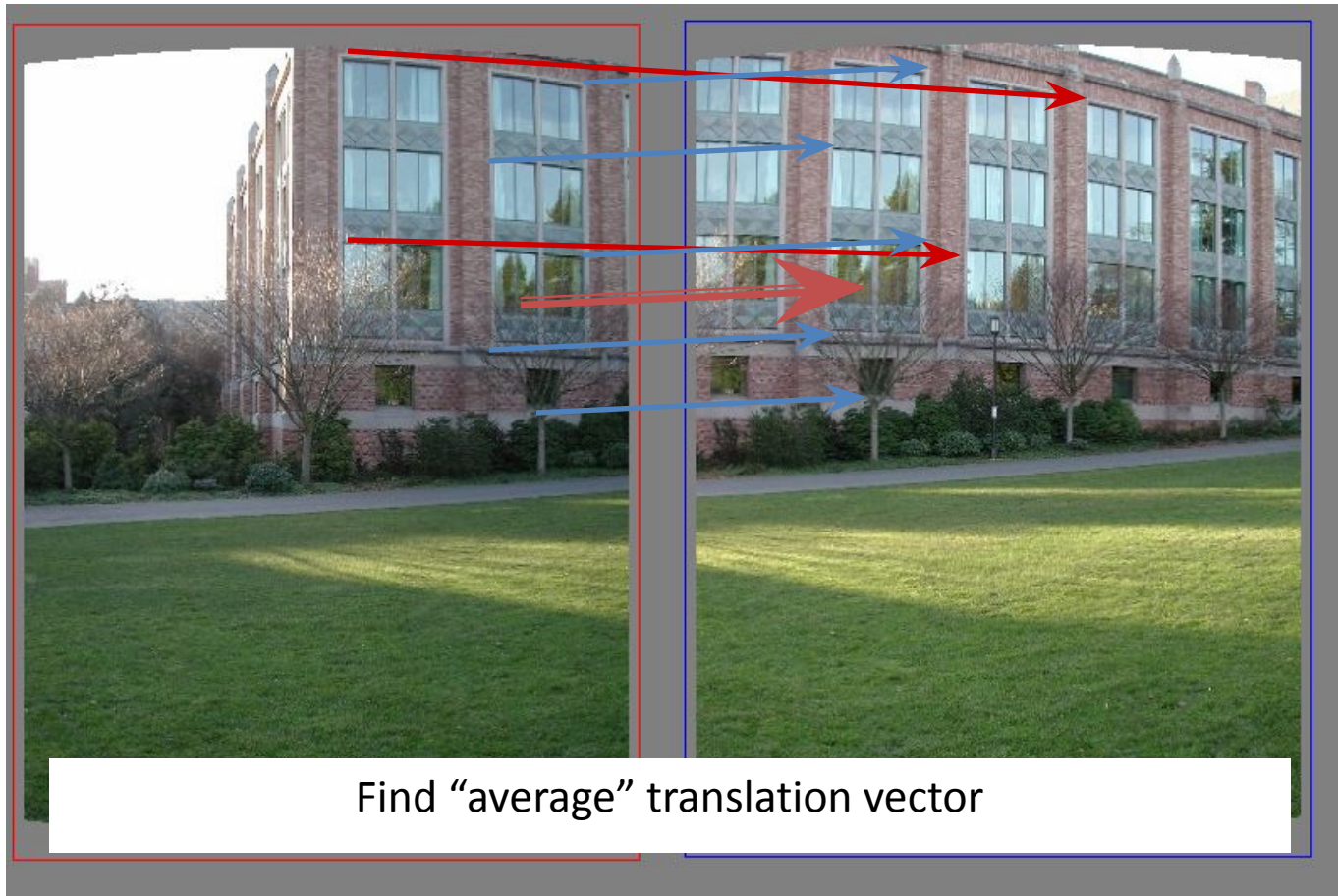
Random Sample Consensus

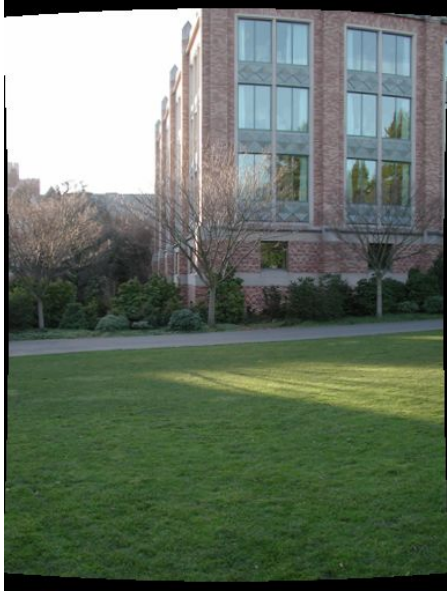


Random Sample Consensus



Least squares fit



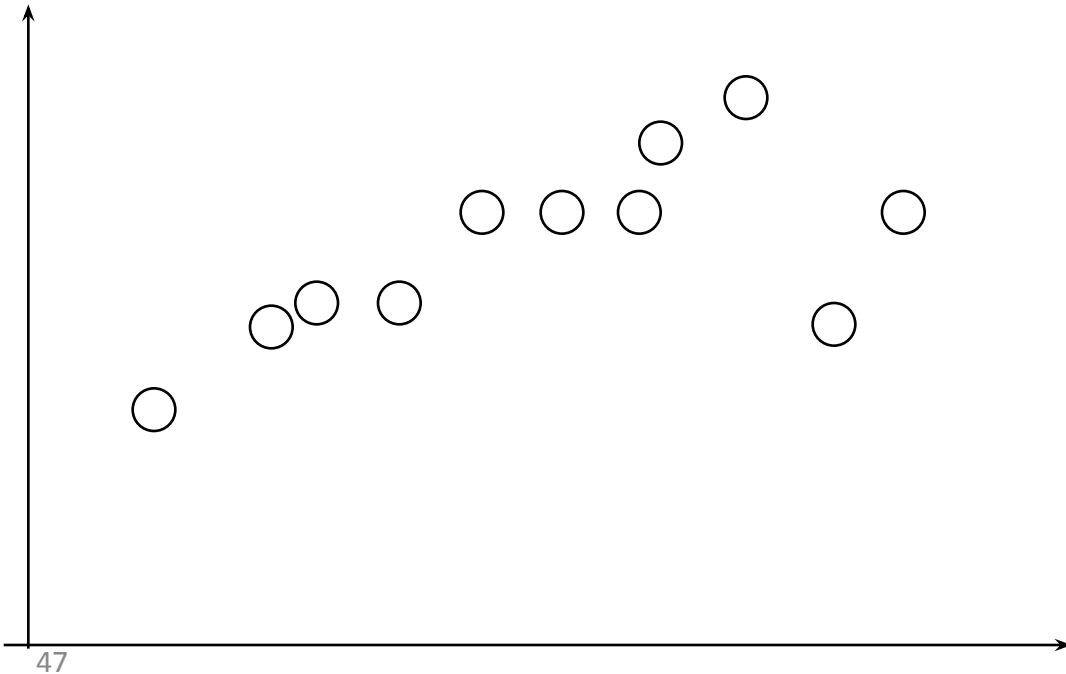


RANSAC for estimating homography

- RANSAC loop:
 1. Select four feature pairs (at random)
 2. Compute homography \mathbf{H} (exact)
 3. Compute inliers where $\|p_i', \mathbf{H} p_i\| < \varepsilon$
 - Keep largest set of inliers
 - Re-compute least-squares \mathbf{H} estimate using all of the inliers

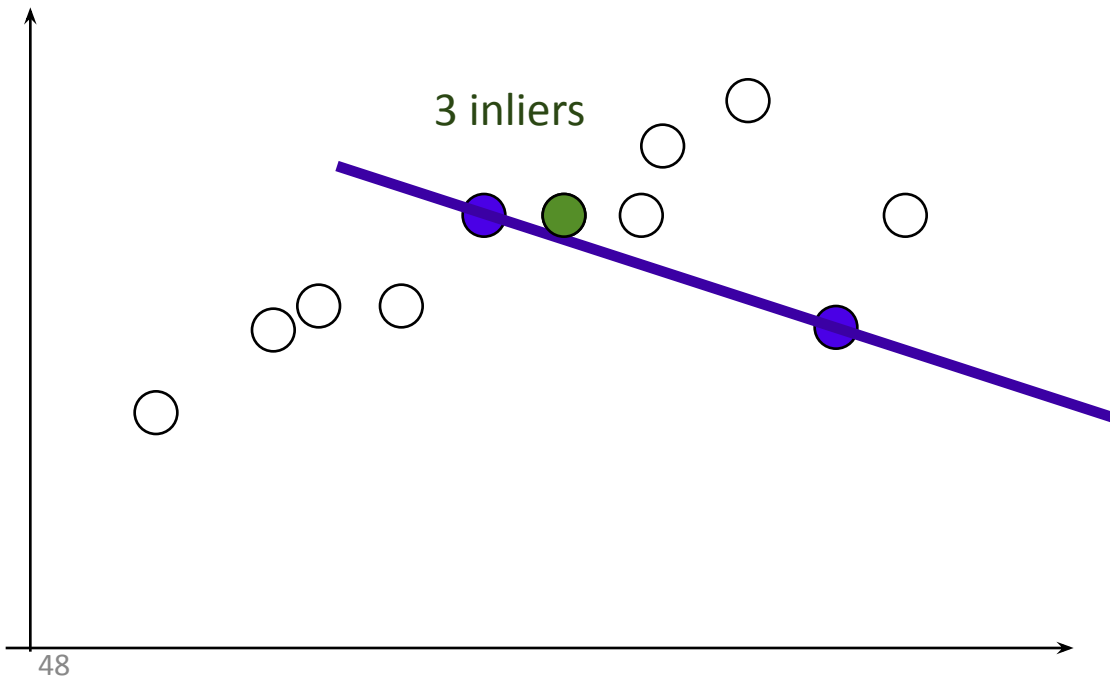
Simple example: fit a line

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



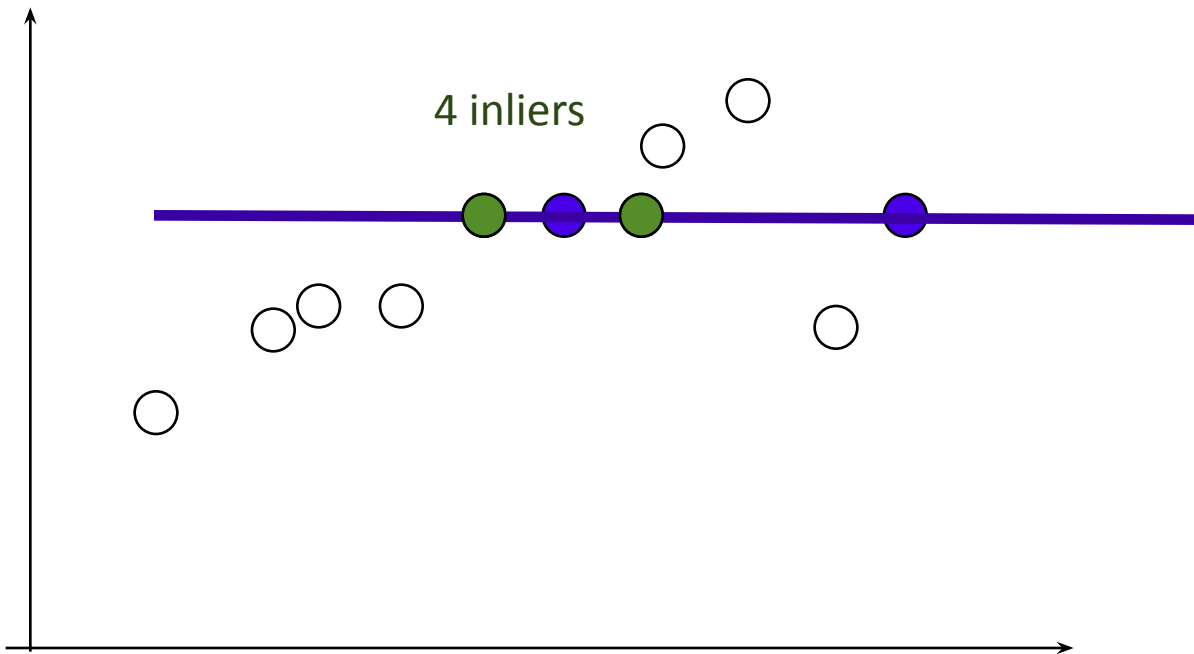
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



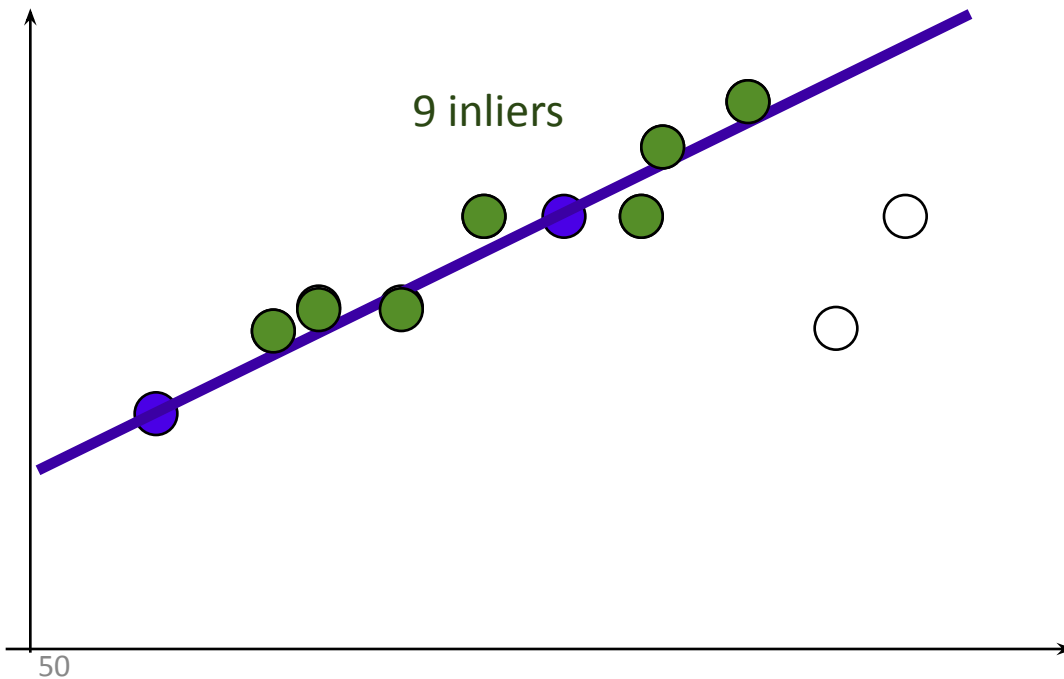
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



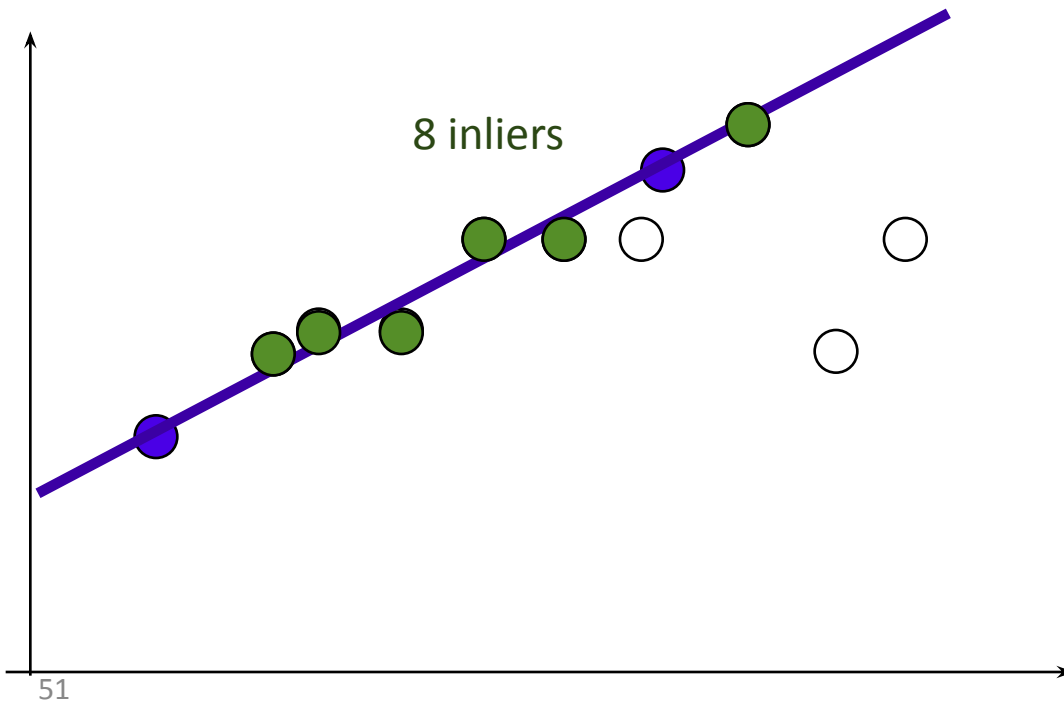
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



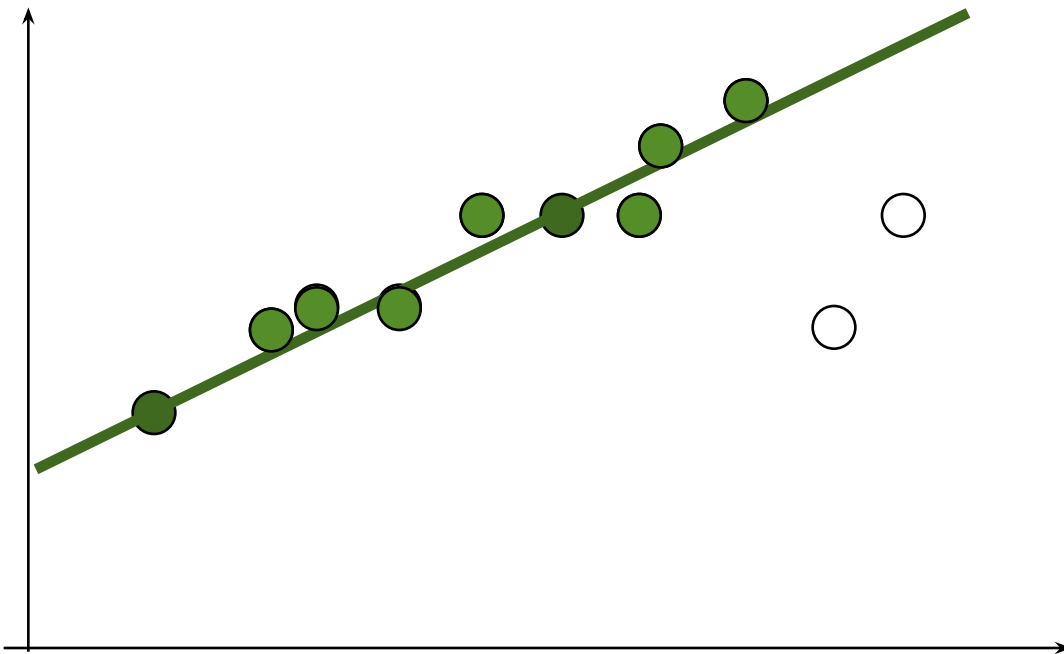
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

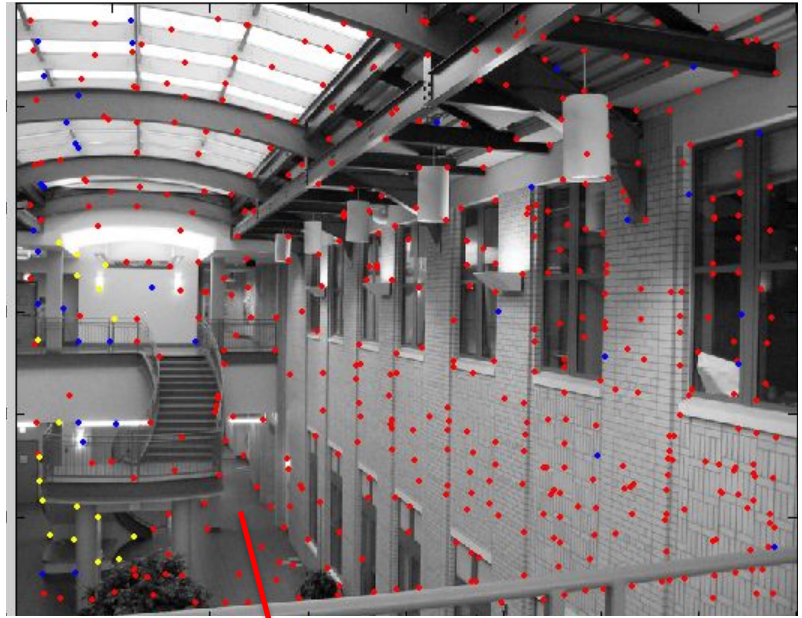
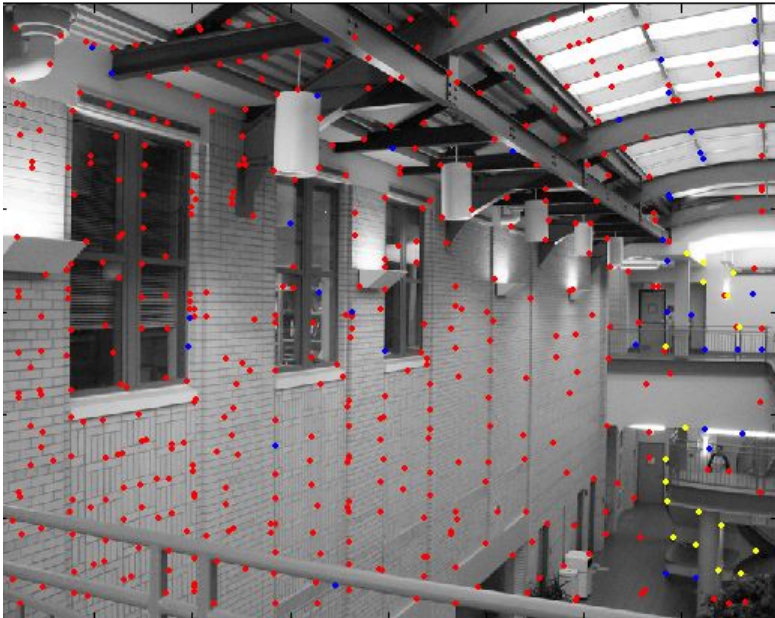


Simple example: fit a line

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



RANSAC



- Red:
rejected by 2nd nearest neighbor criterion
- Blue:
Ransac outliers
- Yellow:
inliers



How many rounds?

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

For probability p of no outliers:

$$N = \log(1 - p) / \log(1 - (1 - \epsilon)^s)$$

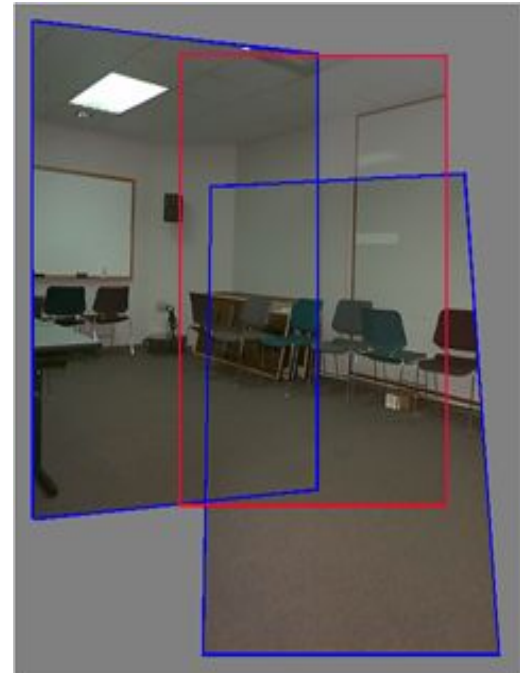
- N , number of samples
- s , size of sample set
- ϵ , proportion of outliers

e.g. for $p = 0.95$

Sample size s	Proportion of outliers ϵ						
	5%	10%	20%	25%	30%	40%	50%
2	2	2	3	4	5	7	11
3	2	3	5	6	8	13	23
4	2	3	6	8	11	22	47
5	3	4	8	12	17	38	95
6	3	4	10	16	24	63	191
7	3	5	13	21	35	106	382
8	3	6	17	29	51	177	766

Rotational mosaics

- Directly optimize rotation and focal length
- Advantages:
 - ability to build full-view panoramas
 - easier to control interactively
 - more stable and accurate estimates



Rotational mosaic

- Projection equations
 1. Project from image to 3D ray
 - $(x_0, y_0, z_0) = (u_0 - u_c, v_0 - v_c, f)$
 2. Rotate the ray by camera motion
 - $(x_1, y_1, z_1) = \mathbf{R}_{01} (x_0, y_0, z_0)$
 3. Project back into new (source) image
 - $(u_1, v_1) = (fx_1/z_1 + u_c, fy_1/z_1 + v_c)$

Computing homography

- Assume we have four matched points: How do we compute homography \mathbf{H} ?

Normalized DLT

1. Normalize coordinates for each image
 - a) Translate for zero mean
 - b) Scale so that average distance to origin is $\sim\sqrt{2}$
3. Compute $\tilde{\mathbf{H}}$ using DLT in normalized coordinates
4. Unnormalize:


$$\mathbf{H} = \mathbf{T}'^{-1} \tilde{\mathbf{H}} \mathbf{T}$$

$$\mathbf{x}'_i = \mathbf{H} \mathbf{x}_i$$

Computing homography

- Assume we have matched points with outliers: How do we compute homography \mathbf{H} ?

Automatic Homography Estimation with RANSAC

1. Choose number of samples N
 2. Choose 4 random potential matches
 3. Compute \mathbf{H} using normalized DLT
 4. Project points from \mathbf{x} to \mathbf{x}' for each potentially matching pair: $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$
 5. Count points with projected distance $< t$
 - E.g., $t = 3$ pixels
 6. Repeat steps 2-5 N times
 - Choose \mathbf{H} with most inliers
- 

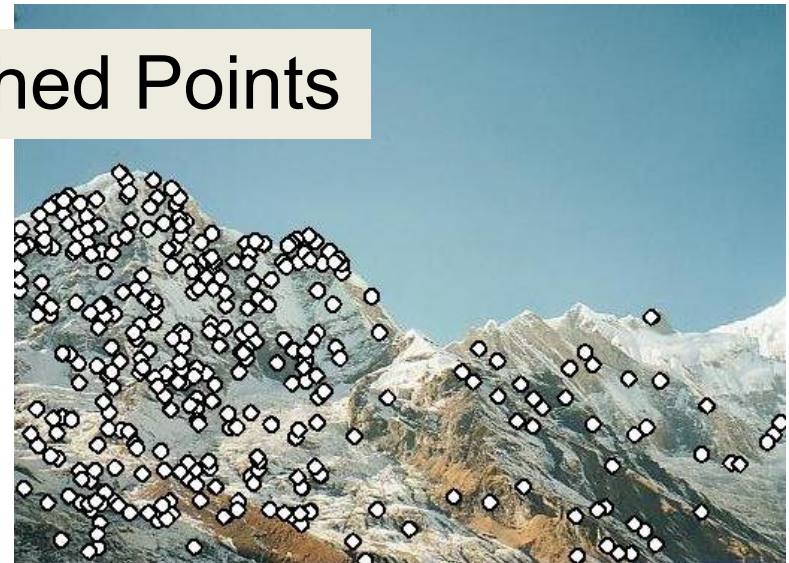
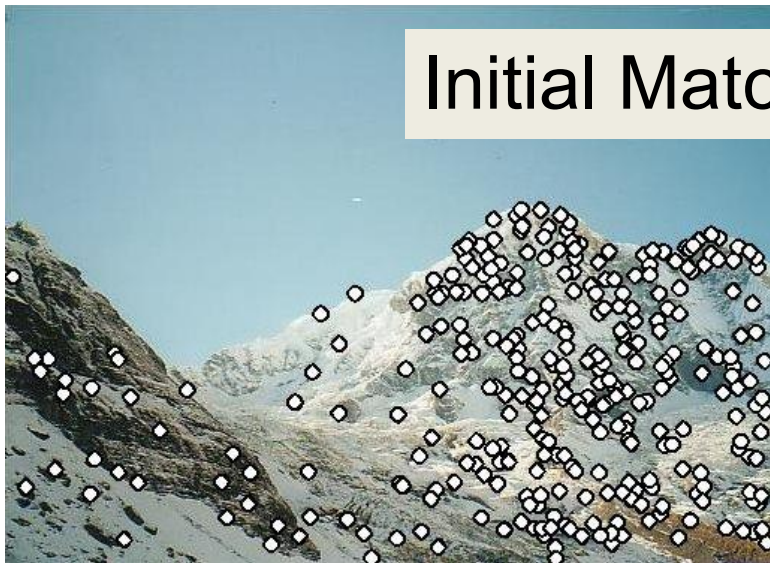
Automatic Image Stitching

1. Compute interest points on each image
1. Find candidate matches
1. Estimate homography **H** using matched points and RANSAC with normalized DLT
1. Project each image onto the same surface and blend

RANSAC for Homography



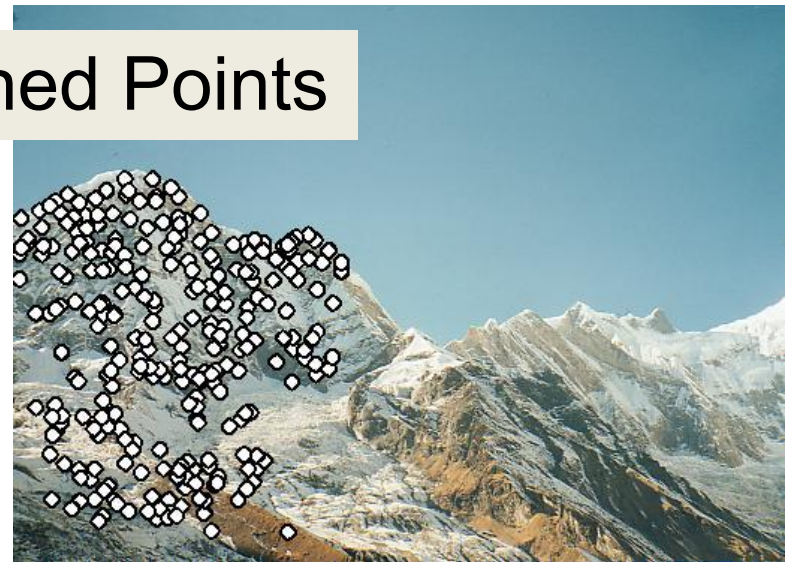
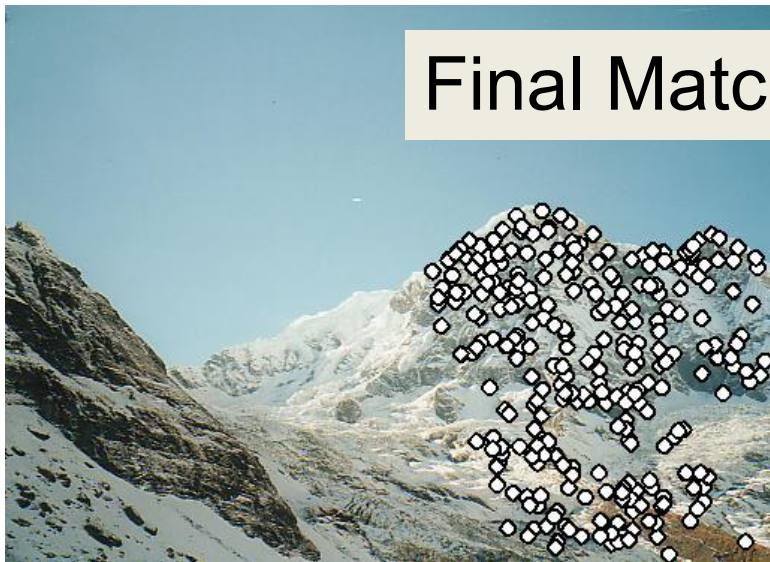
Initial Matched Points



RANSAC for Homography



Final Matched Points



RANSAC for Homography

