

Image alignment and RANSAC

<Computer Vision>

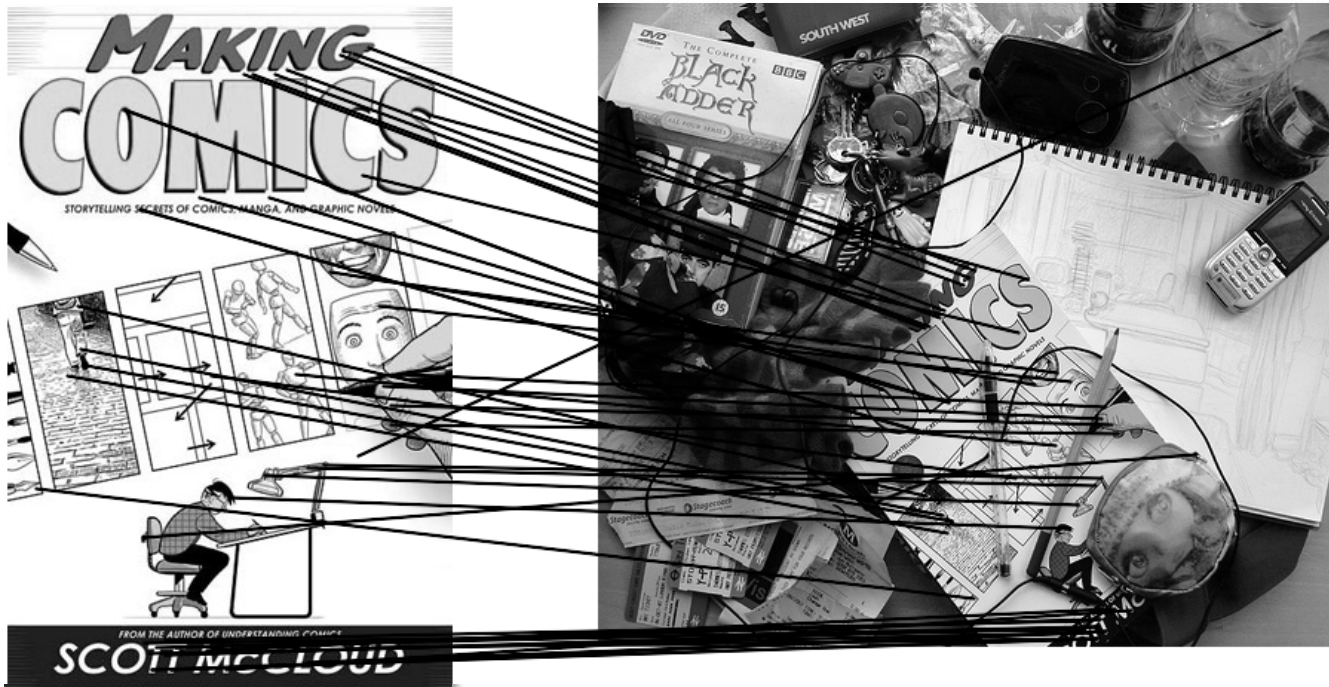
Department of Robot Engineering

Prof. Younggun Cho



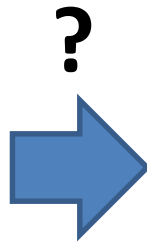
Computing transformations

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



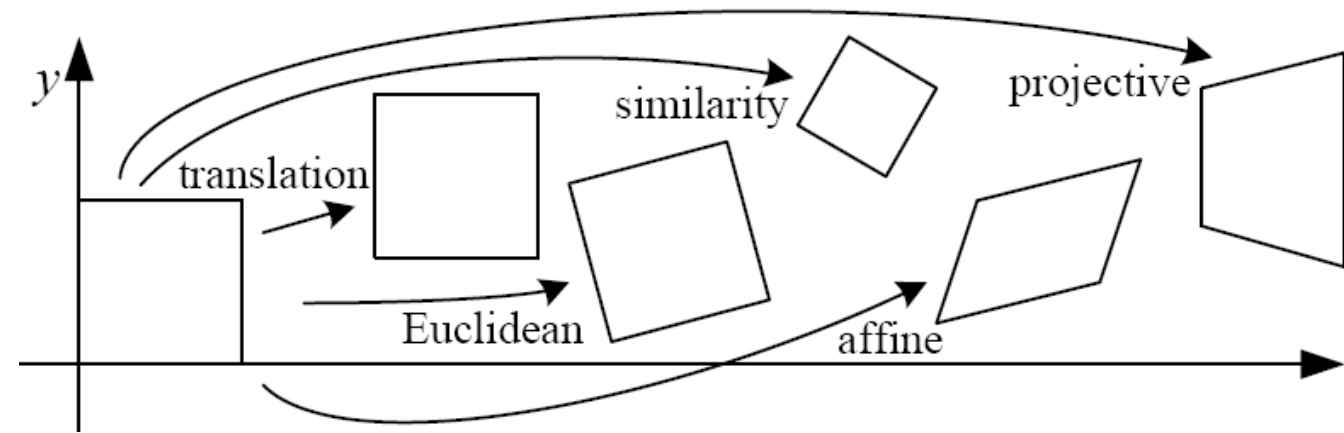
- Find transform T that best “agrees” with the matches

Computing transformations



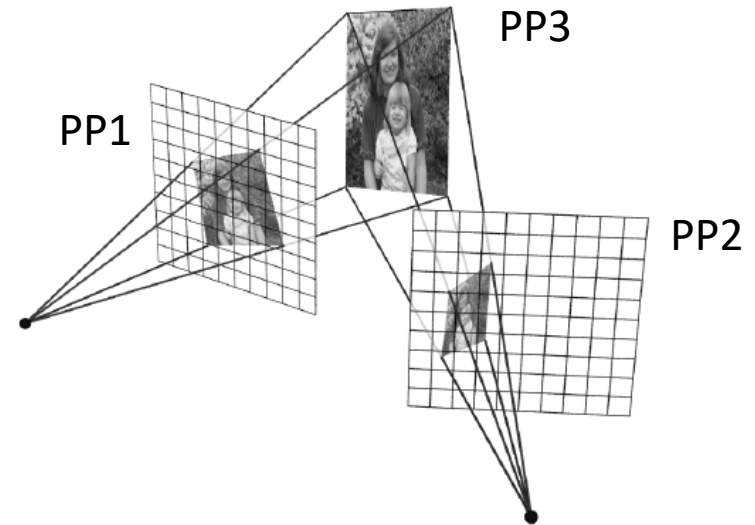
Back to warping: image homographies

Classification of 2D transformations



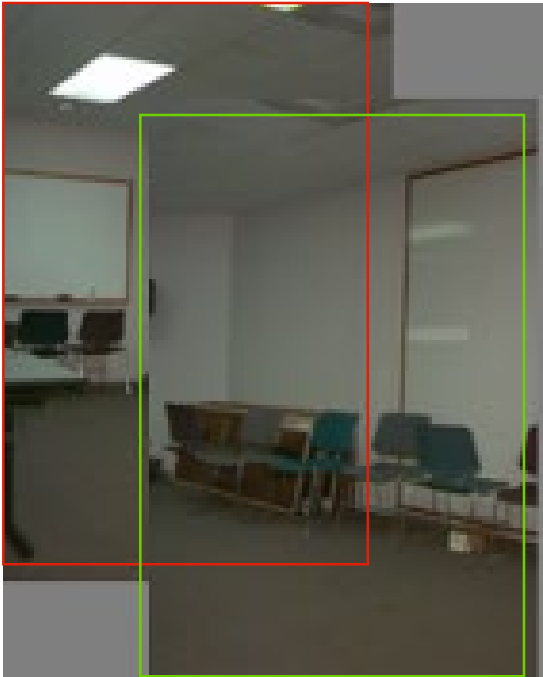
Which kind transformation is needed to warp projective plane 1 into projective plane 2?

- A projective transformation (a.k.a. a homography).

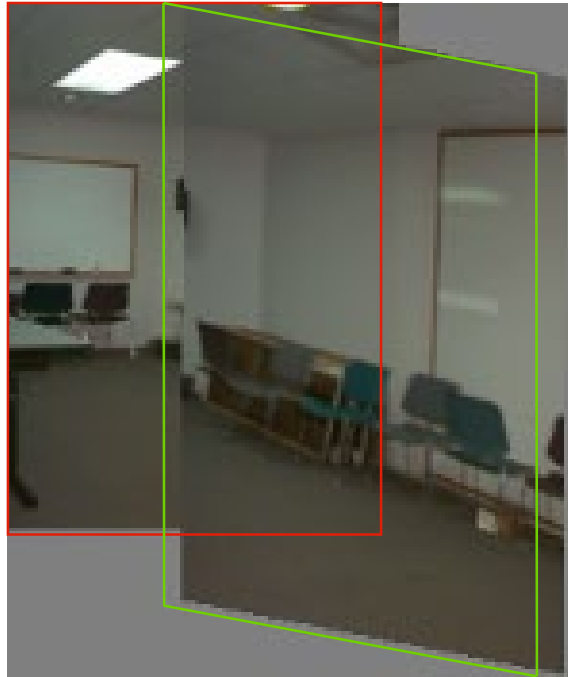


Warping with different transformations

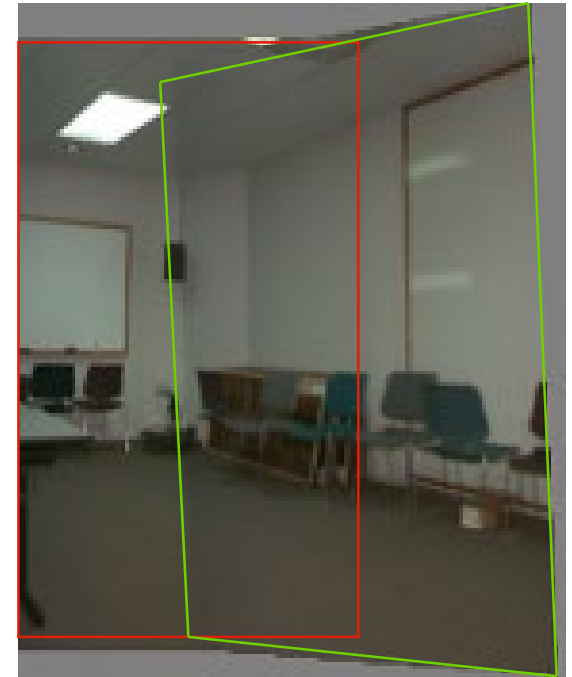
translation



affine



pProjective (homography)



When can we use homographies?

We can use homographies when...

1. ... the scene is planar;
or



2. ... the scene is very far or has small (relative) depth variation
→ scene is approximately planar



We can use homographies when...

3. ... the scene is captured under camera rotation only (no translation or pose change)



More on why this is the case in a later lecture.

Computing with homographies

Applying a homography

1. Convert to homogeneous coordinates:

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

What is the size of the homography matrix?

Answer: 3 x 3

2. Multiply by the homography matrix:

$$P' = H \cdot P$$

How many degrees of freedom does the homography matrix have?

3. Convert back to heterogeneous coordinates:

$$P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} \Rightarrow p' = \begin{bmatrix} x'/w' \\ y'/w' \end{bmatrix}$$

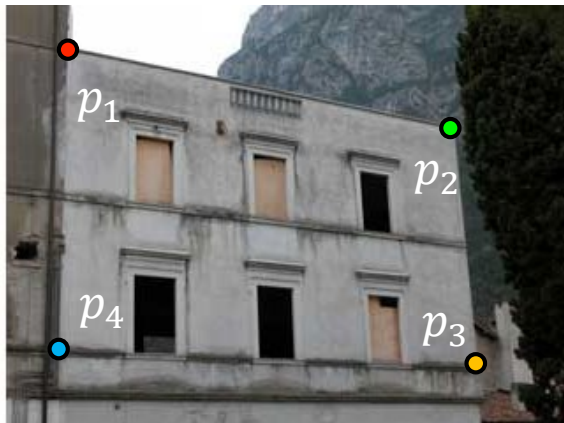
How can we compute homography matrix?

The direct linear transform (DLT)

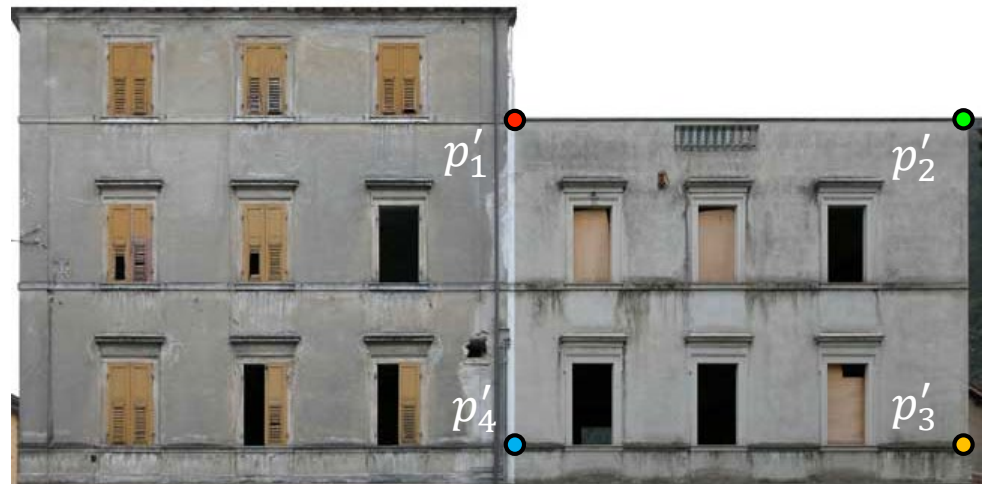
Create point correspondences

Given a set of matched feature points $\{p_i, p'_i\}$ find the best estimate of H such that

$$P' = H \cdot P$$



original image



target image

How many correspondences do we need?

Determining the homography matrix

Write out linear equation for each correspondence:

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

Expand matrix multiplication:

$$x' = \alpha(h_1x + h_2y + h_3)$$

$$y' = \alpha(h_4x + h_5y + h_6)$$

$$1 = \alpha(h_7x + h_8y + h_9)$$

Divide out unknown scale factor:

$$x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3)$$


$$y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6)$$

*How do you
rearrange terms
to make it a
linear system?*

$$x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3)$$

$$y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6)$$

Just rearrange the terms



$$h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0$$

$$h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0$$

Determining the homography matrix

Re-arrange terms:

$$h_7xx' + h_8yx' + h_9x' - h_1x - h_2y - h_3 = 0$$

$$h_7xy' + h_8yy' + h_9y' - h_4x - h_5y - h_6 = 0$$

Re-write in matrix form:

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

*How many equations
from one point
correspondence?*

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

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

Determining the homography matrix

Stack together constraints from multiple point correspondences:

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

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \\ -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \\ \vdots & & & & & & & & \\ -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

Reminder: Determining affine transformations

Affine transformation:

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

Vectorize
transformation
parameters:
Stack equations
from point
correspondences:

$$\begin{bmatrix} x' \\ y' \\ x' \\ y' \\ \vdots \\ x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ \vdots & & & \vdots & & \\ x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{bmatrix}$$

Notation in system form:

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

Reminder: Determining affine transformations

Convert the system to a linear least-squares problem:

$$E_{\text{LLS}} = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

Expand the error:

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

Minimize the error:

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

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

In Matlab:

$$\mathbf{x} = \mathbf{A} \setminus \mathbf{b}$$

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

Determining the homography matrix

Stack together constraints from multiple point correspondences:

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

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \\ -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \\ \vdots & & & & & & & & \\ -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Homogeneous linear least squares problem

- How do we solve this?
- Solve with SVD

Singular Value Decomposition


$$\begin{aligned} \mathbf{A} &= \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \\ \text{ortho-normal} \quad & \text{diagonal} \quad \text{ortho-normal} \quad \text{unit norm constraint} \\ \text{n} \times \text{m} \quad & \text{n} \times \text{n} \quad \text{n} \times \text{m} \quad \text{m} \times \text{m} \end{aligned}$$
$$= \sum_{i=1}^9 \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

$\text{n} \times 1 \quad 1 \times \text{m}$

General form of total least squares

(Warning: change of notation. \mathbf{x} is a vector of parameters!)

$$\begin{aligned} E_{\text{TLS}} &= \sum_i (\mathbf{a}_i \mathbf{x})^2 \\ &= \|\mathbf{A}\mathbf{x}\|^2 && \text{(matrix form)} \\ \|\mathbf{x}\|^2 &= 1 && \text{constraint} \end{aligned}$$

minimize	$\ \mathbf{A}\mathbf{x}\ ^2$		(Rayleigh quotient)
			
subject to	$\ \mathbf{x}\ ^2 = 1$	minimize	$\frac{\ \mathbf{A}\mathbf{x}\ ^2}{\ \mathbf{x}\ ^2}$

Solution is the eigenvector
corresponding to smallest
eigenvalue of

$$\mathbf{A}^\top \mathbf{A}$$

(equivalent)

Solution is the column of \mathbf{V}
corresponding to smallest
singular value

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

Solving for H using DLT

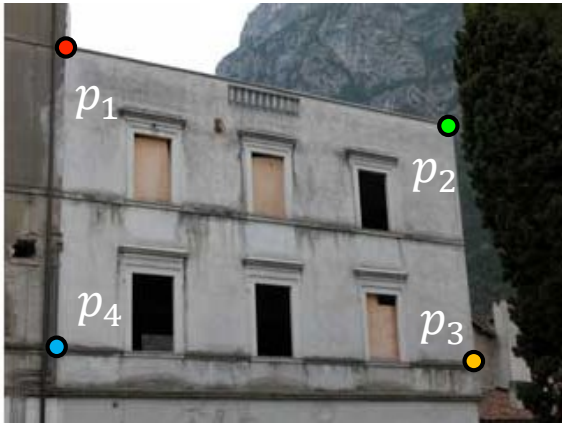
Given $\{x_i, x'_i\}$ solve for H such that $x' = Hx$

1. For each correspondence, create 2x9 matrix A_i
2. Concatenate into single $2n \times 9$ matrix A
3. Compute SVD of $A = U\Sigma V^T$
4. Store singular vector of the smallest singular value $h = v_{\hat{i}}$
5. Reshape to get H

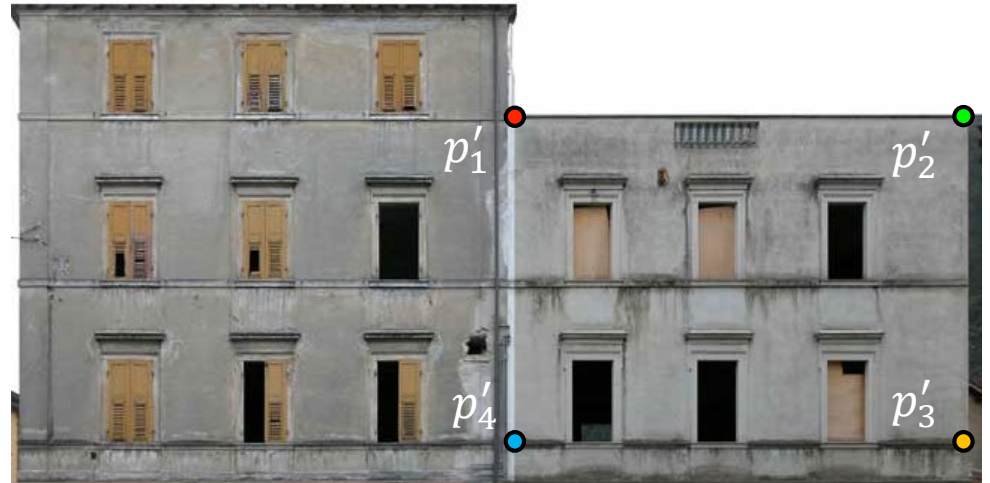
Linear least squares estimation only works when the transform function is **linear! (duh)**

Also doesn't deal well with **outliers**.

Create point correspondences



original image



target image

How do we automate this step?

The image correspondence pipeline

1. Feature point detection

- Detect corners using the Harris corner detector.

2. Feature point description

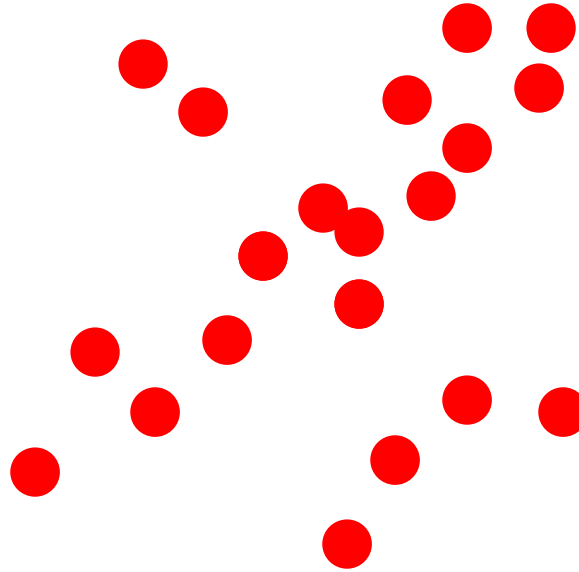
- Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching



Random Sample Consensus (RANSAC)

Fitting lines
(with outliers)

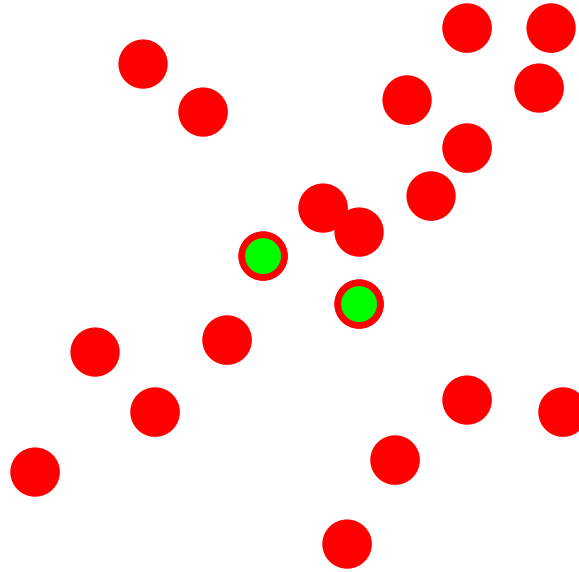


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

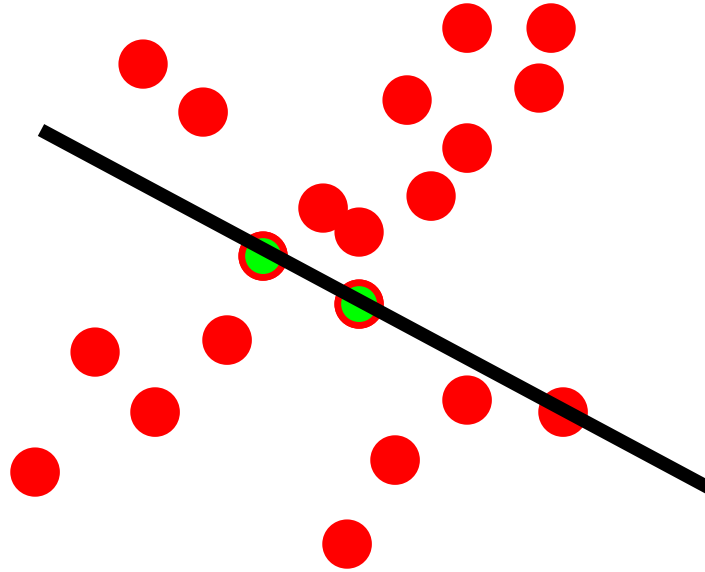


Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



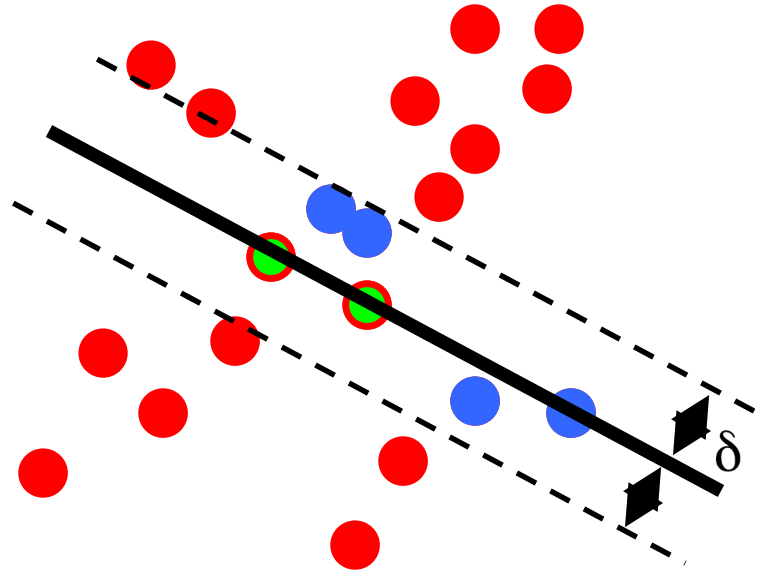
Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

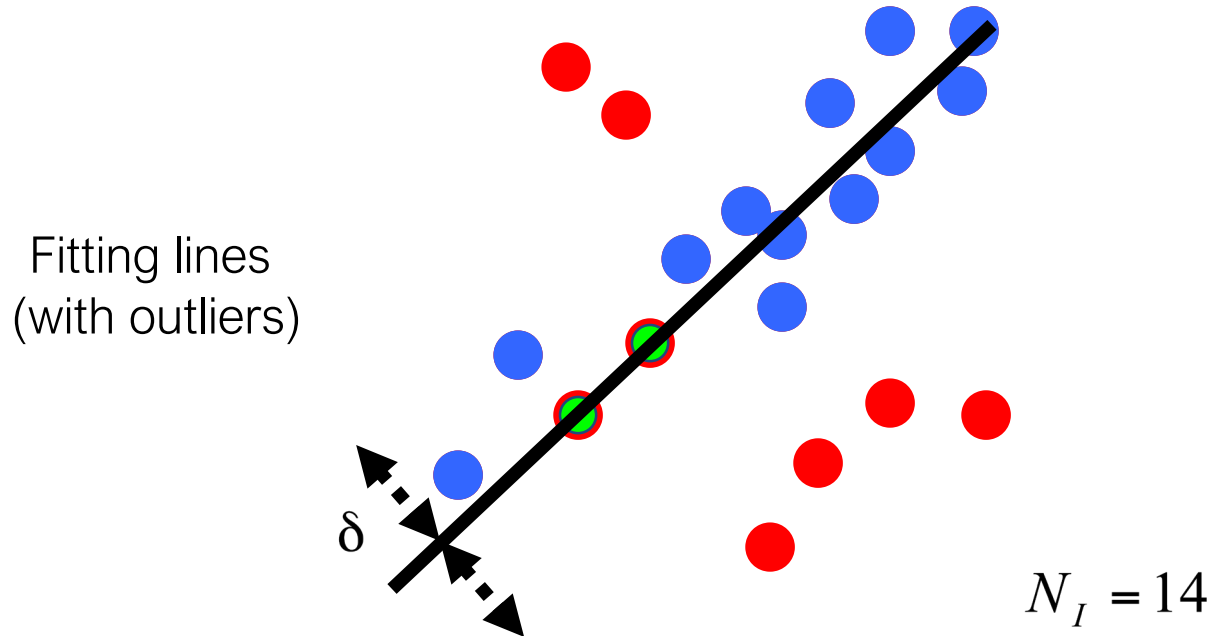
$$N_I = 6$$



Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence



Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

- Number of samples N
 - Choose N so that, with probability p , at least one random sample is free from outliers (e.g. $p=0.99$) (outlier ratio: e)
- Number of sampled points s
 - Minimum number needed to fit the model
- Distance threshold δ
 - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
 - Zero-mean Gaussian noise with std. dev. σ : $t^2=3.84\sigma^2$

$$N = \frac{\log(1 - p)}{\log\left(1 - (1 - e)^s\right)}$$

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

Given two images...



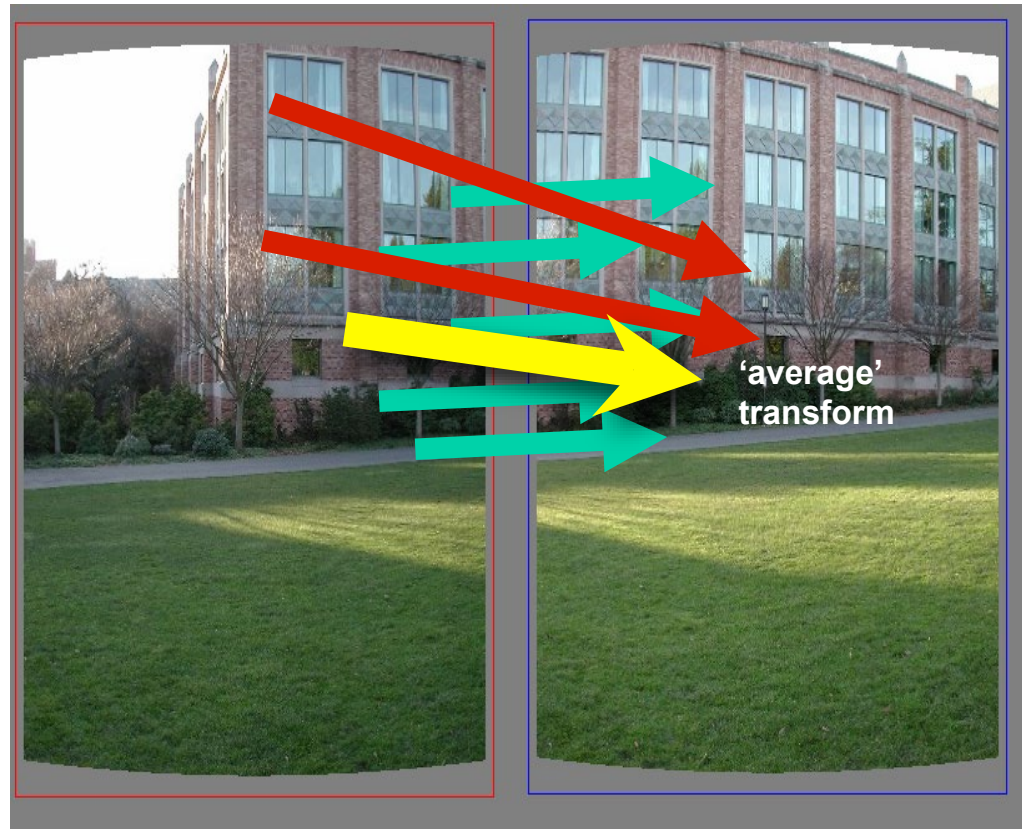
find matching features (e.g., SIFT) and a translation transform

Matched points will usually contain bad correspondences



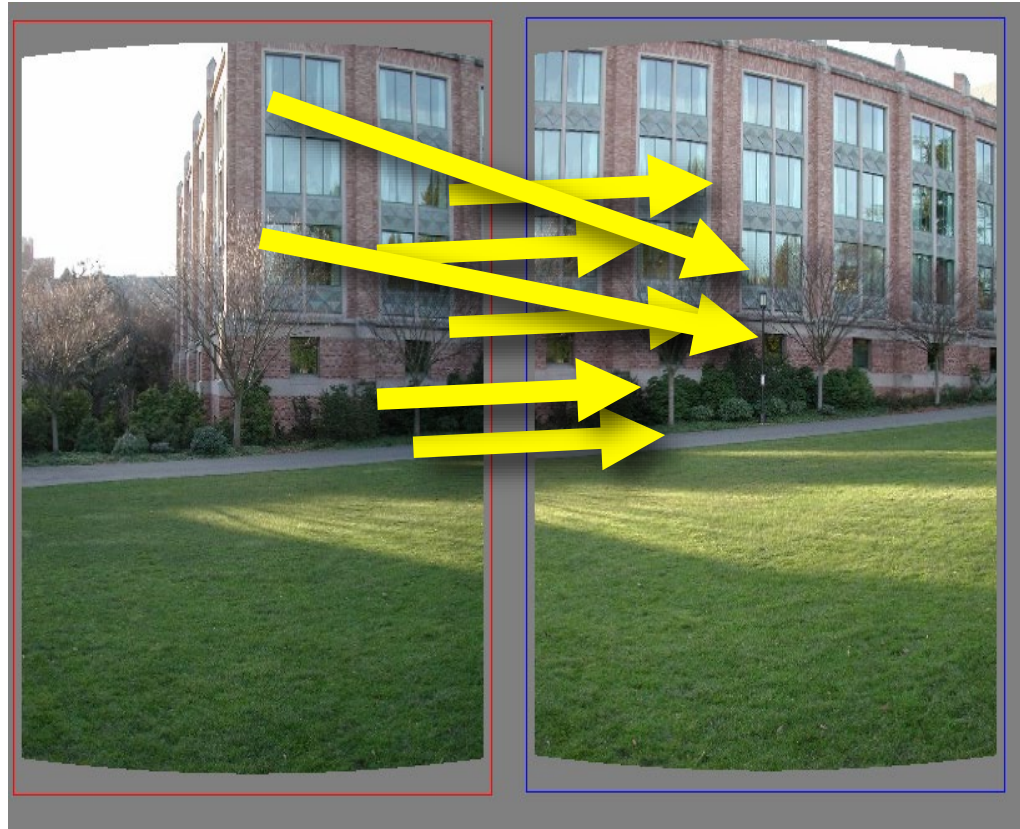
how should we estimate the transform?

LLS will find the 'average' transform

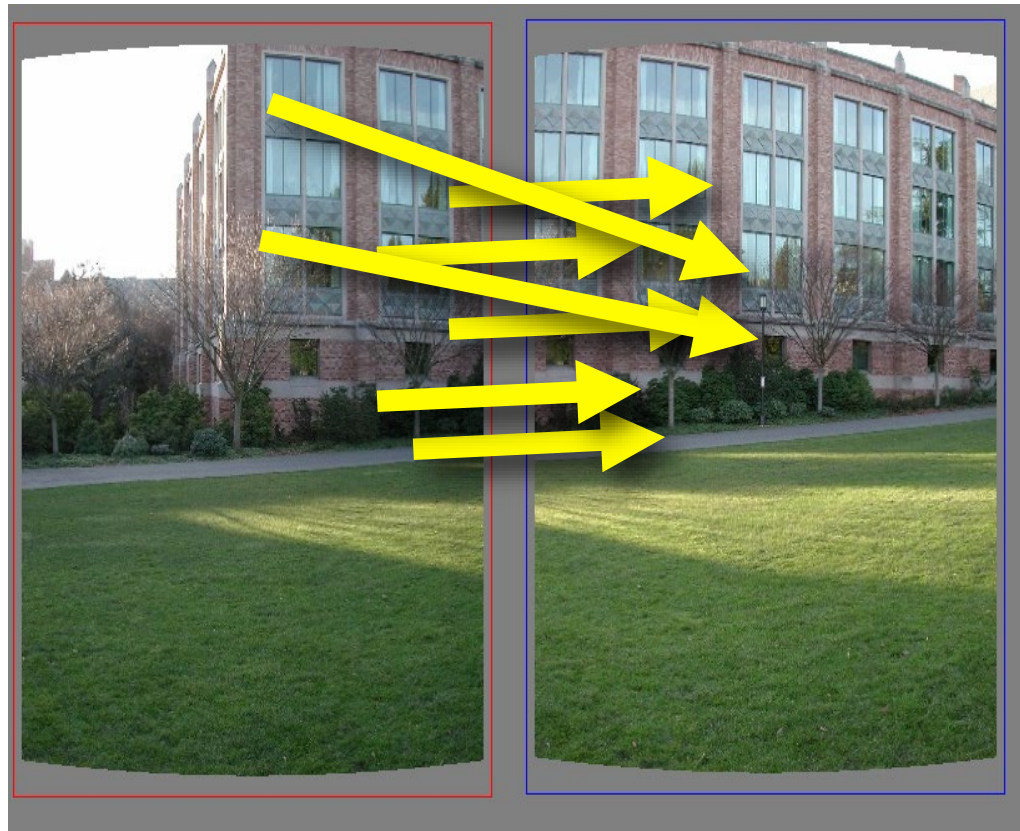


solution is corrupted by bad correspondences

Use RANSAC

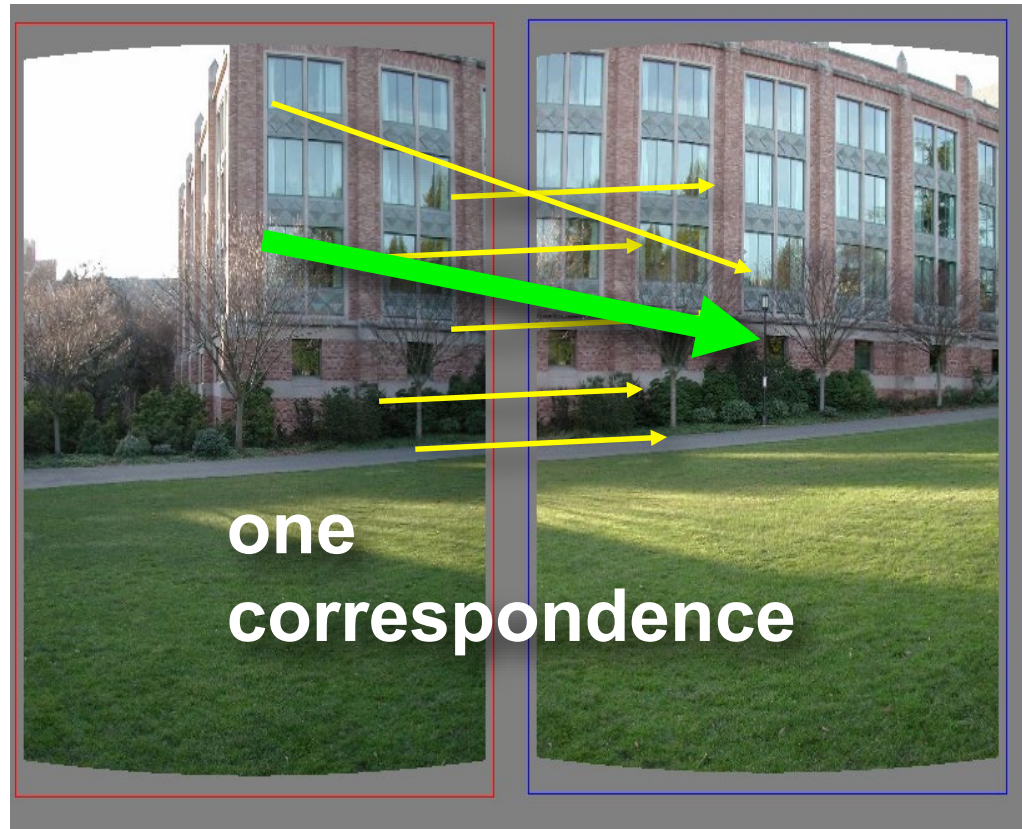


How many correspondences to compute translation transform?

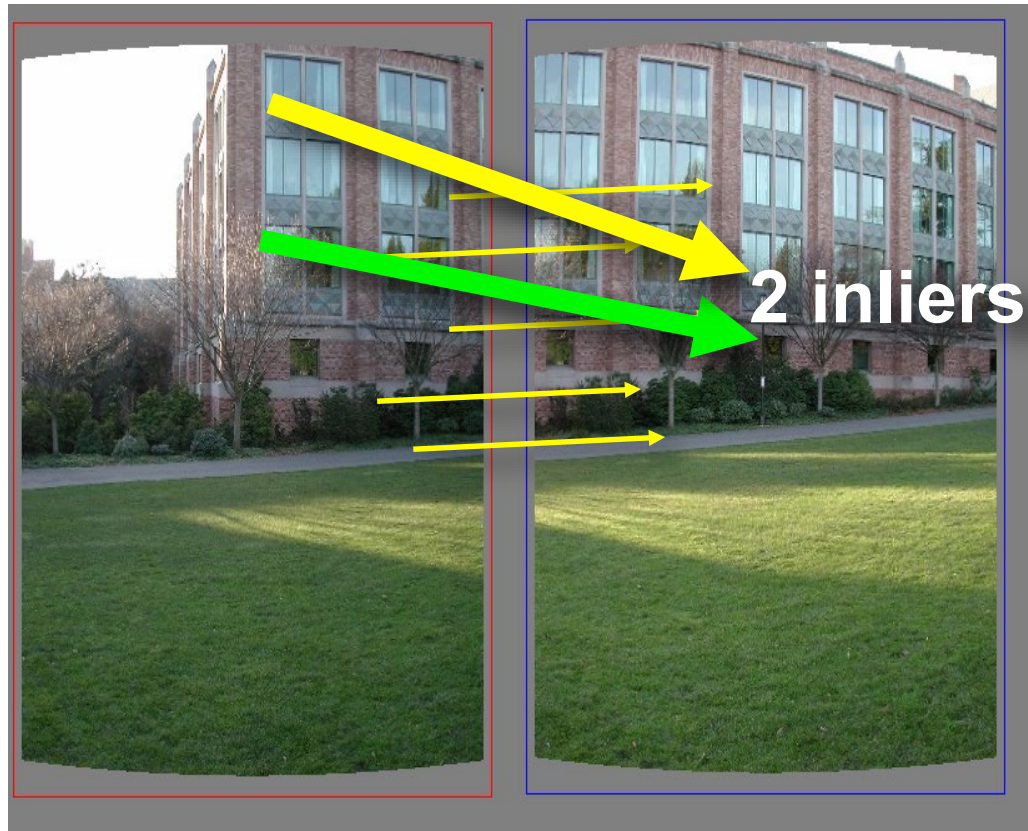


Need only **one correspondence**, to find translation model

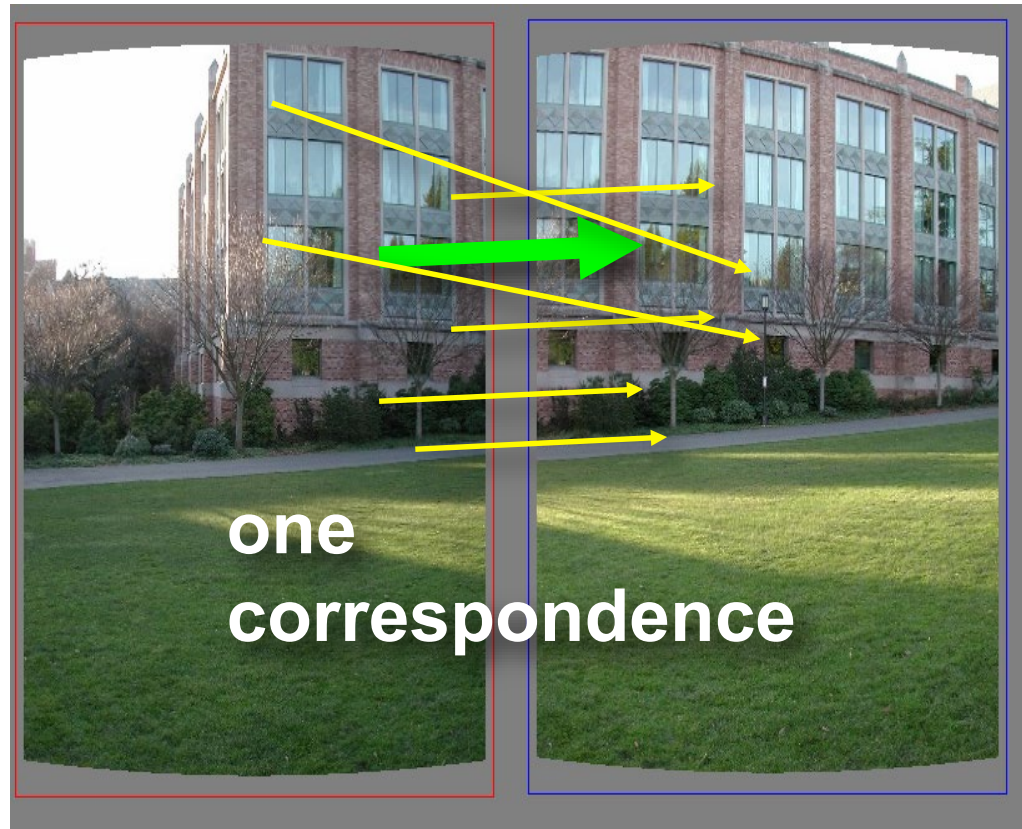
Pick one correspondence, count inliers



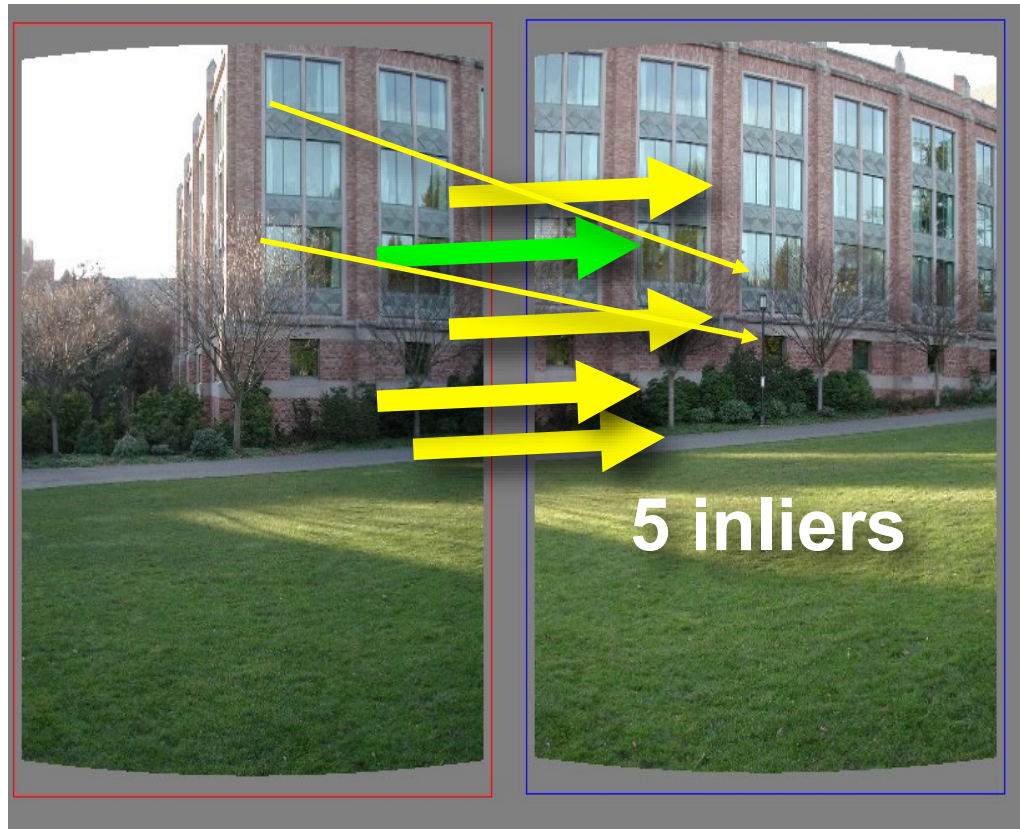
Pick one correspondence, count inliers



Pick one correspondence, count inliers




Pick one correspondence, count inliers

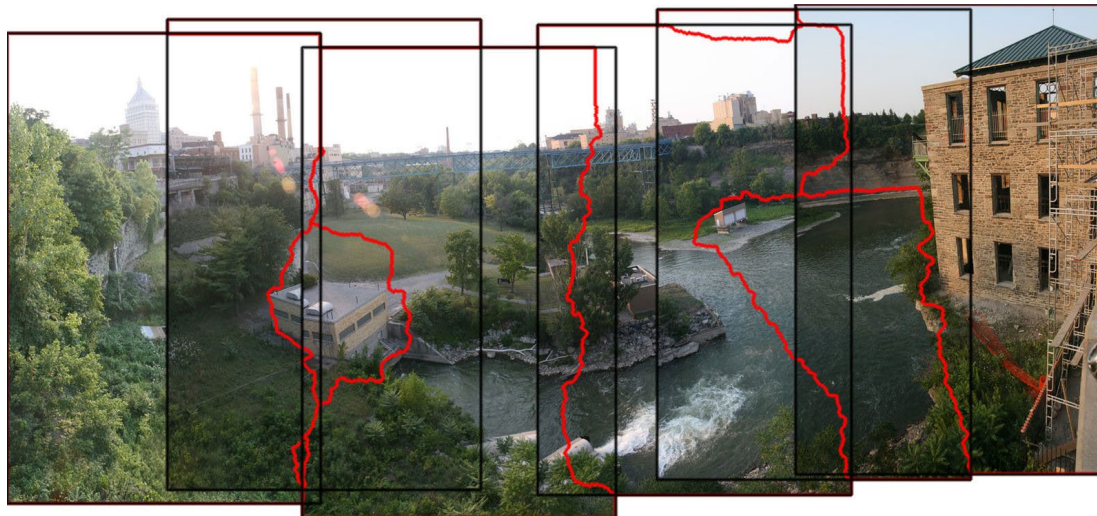


Pick the model with the highest number of inliers!

Estimating homography using RANSAC

- RANSAC loop
 1. Get four point correspondences (randomly)
 2. Compute H using DLT
 3. Count inliers
 4. Keep H if largest number of inliers
 - Recompute H using all inliers
- 

Useful for...





The image correspondence pipeline

1. Feature point detection

- Detect corners using the Harris corner detector.

2. Feature point description

- Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching *and* homography estimation

- Do both simultaneously using RANSAC.

Lab. RANSAC-based Matching

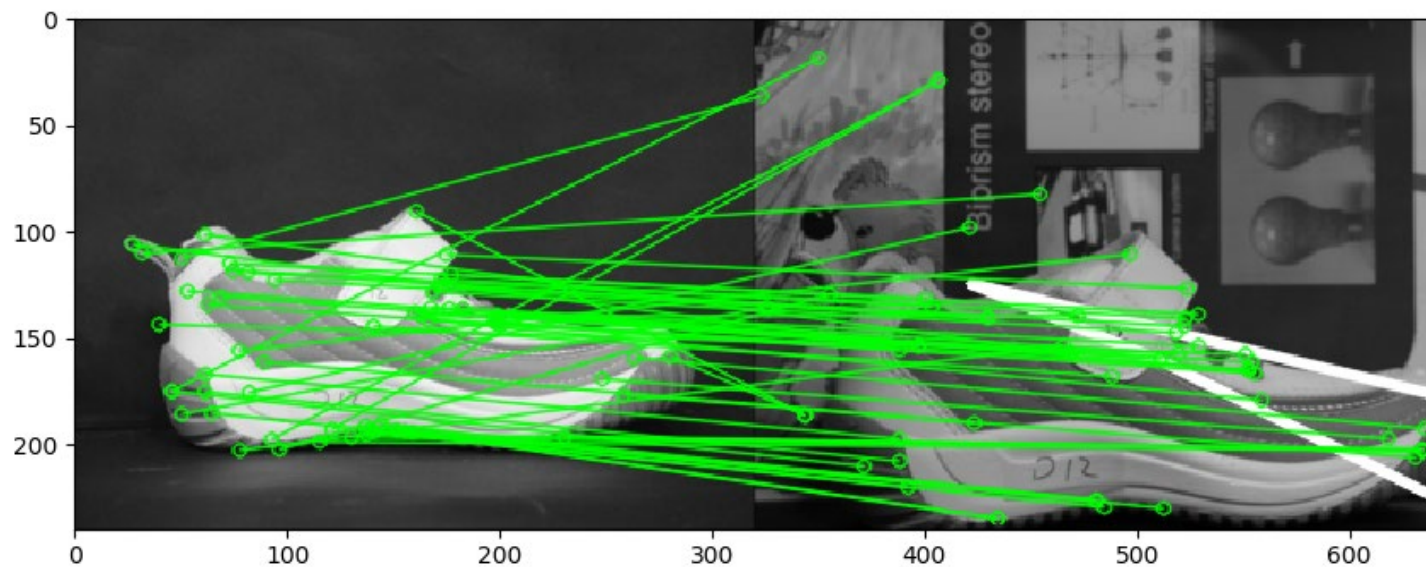
```
1 import numpy as np
2 import cv2
3 from matplotlib import pyplot as plt
4
5 MIN_MATCH_COUNT = 4
6
7 img1 = cv2.imread('View1.bmp',0) # queryImage
8 img2 = cv2.imread('View2.bmp',0) # trainImage
9
10 # Initiate SIFT detector
11 sift = cv2.xfeatures2d.SIFT_create()
12
13 # find the keypoints and descriptors with SIFT
14 kp1, des1 = sift.detectAndCompute(img1,None)
15 kp2, des2 = sift.detectAndCompute(img2,None)
16
17 FLANN_INDEX_KDTREE = 0
18 index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5) #(key=value)
19 search_params = dict(checks = 50)
20
21 flann = cv2.FlannBasedMatcher(index_params, search_params)
22
23 matches = flann.knnMatch(des1,des2,k=2)
24
```



```
25 # store all the good matches as per Lowe's ratio test.
26 good = []
27 for m,n in matches:
28     if m.distance < 0.9*n.distance:
29         good.append(m)
30
31 if len(good)>MIN_MATCH_COUNT:
32     src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2) # -1: auto size
33     dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)
34
35     M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
36     #M, mask = cv2.findHomography(src_pts, dst_pts)
37
38     matchesMask = mask.ravel().tolist()
39
40     h,w = img1.shape
41     pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
42     dst = cv2.perspectiveTransform(pts,M)
43
44     img2 = cv2.polylines(img2,[np.int32(dst)],True,255,3, cv2.LINE_AA)
45 else:
46     print("Not enough matches are found - %d/%d" % (len(good),MIN_MATCH_COUNT))
47     matchesMask = None
48
```

```
49 draw_params = dict(matchColor = (0,255,0), # draw matches in green color
50                      singlePointColor = None,
51                      matchesMask = matchesMask, # draw only inliers
52                      flags = 2)
53
54 img3 = cv2.drawMatches(img1,kp1,img2,kp2,good,None,**draw_params)
55
56 plt.imshow(img3, 'gray'),plt.show()
```

W/O RANSAC



W/ RANSAC

