CSCI 497P/597P: Computer Vision

Scott Wehrwein

RANSAC: Fitting Transforms with Outliers

Reading

- Szeliski: Chapter 6.1

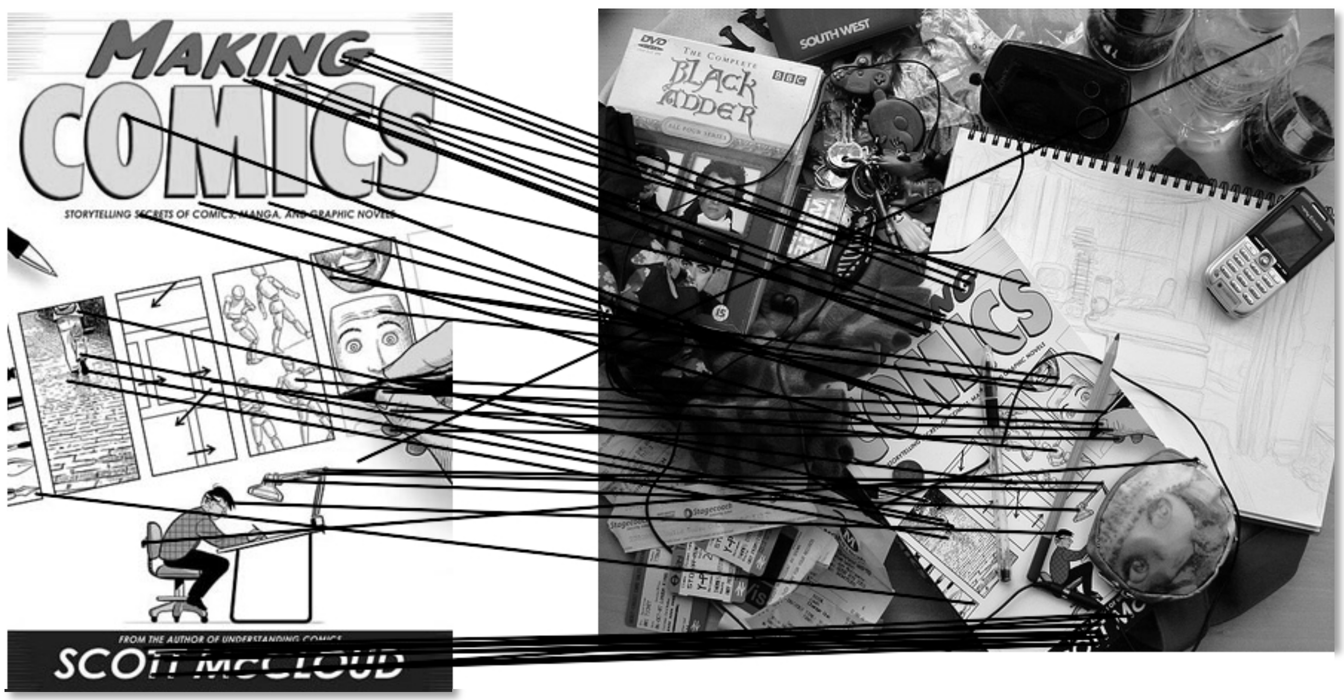
Goals

- Understand the Random Sample Consensus (RANSAC) algorithm.
- Be prepared to implement RANSAC for fitting image coordinate transforms using matches that may contain outliers.

Announcements

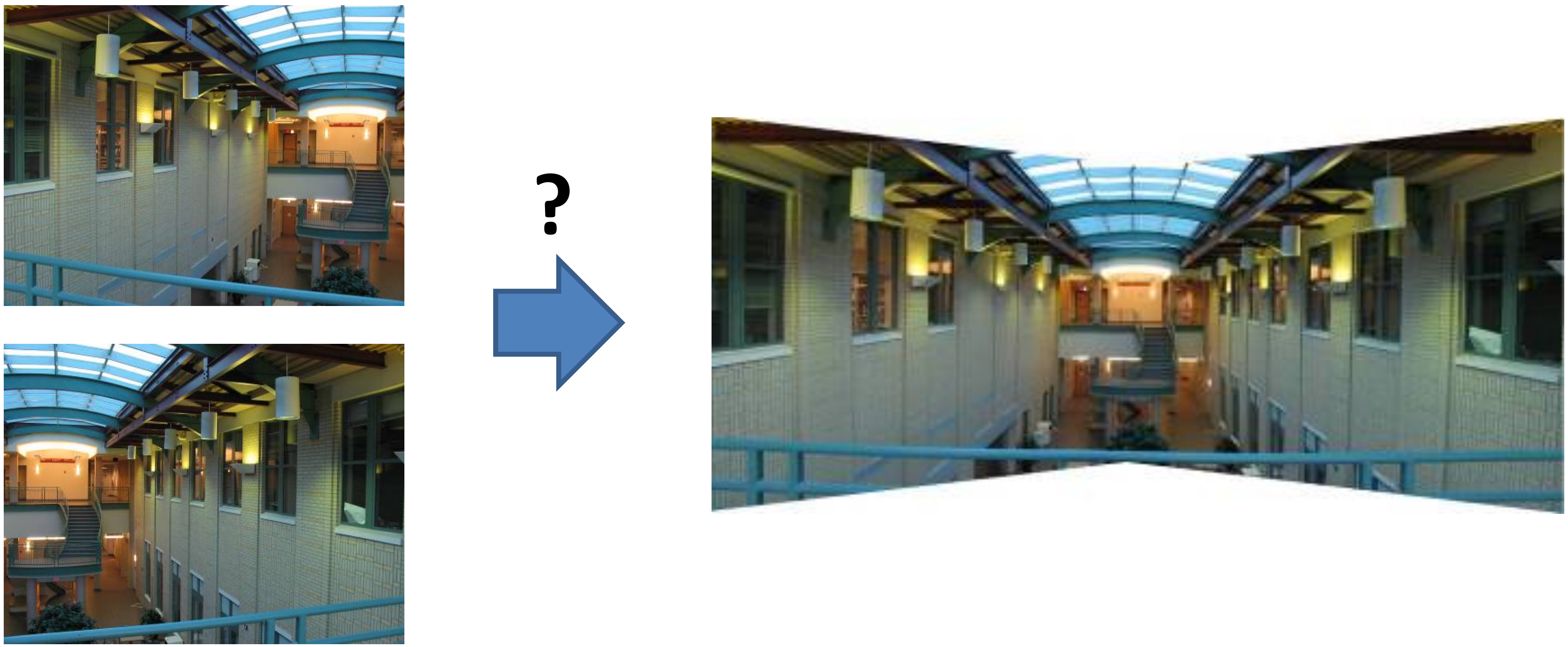
Computing transformations

- 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



Fitting a Homography: TL;DM

- For each feature match $(x_i, y_i) \text{ -> } (x_i', y_i')$, fill in 2 rows of A:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}$$

Fitting a Homography: TL;DM

- For each feature match (x_i, y_i) -> (x_i', y_i'), fill in 2 rows of A
- Solve the homogeneous least squares problem $\min_h ||Ah||^2$:
  - Take the SVD of A to get U, S, and V.
  - Let h be the right singular vector of A whose singular value is smallest.
  - Let h be the column of V (row of V^T) whose column index is the same as that of the smallest diagonal entry of S.

Solving for homographies

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

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

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

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

Recap: Two Common Optimization Problems

Problem statement	Solution
minimize $\lVert \mathbf{Ax} - \mathbf{b} \rVert^2$ least squares solution to $\mathbf{Ax} = \mathbf{b}$	$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$ `np.linalg.lstsq(A, b)`

Problem statement	Solution
minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}$ s.t. $\mathbf{x}^T\mathbf{x} = 1$ non - trivial lsq solution to $\mathbf{Ax} = 0$	$U, \Sigma, V = \mathrm{svd}(A)$ $x \leftarrow v_{\arg\min_i \Sigma[i,i]}$ `U, s, V = np.linalg.svd(A)`

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?

Code: fitting affine transformations

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

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

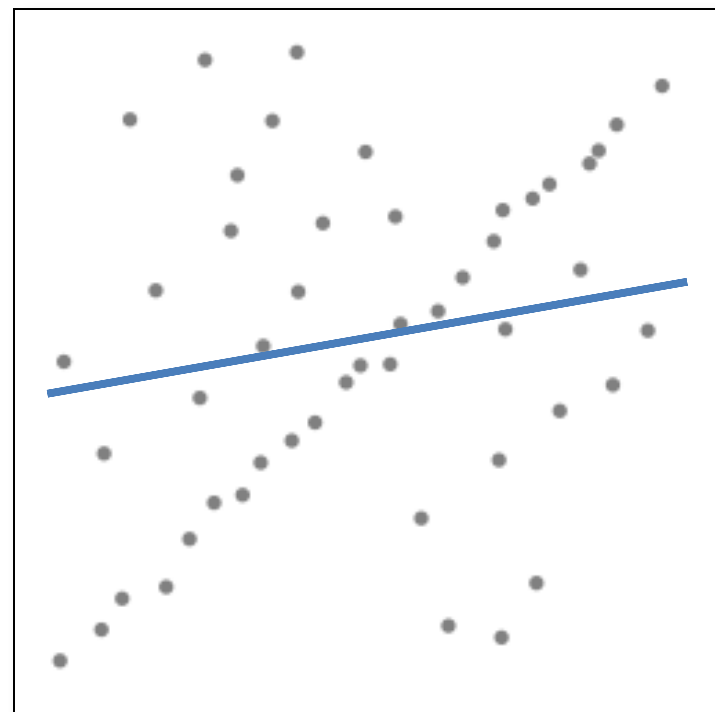
Outliers


outliers
inliers
MAKING COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
DVD
THE COMPLETE BLACK ADDER
BBC
SOUTH WEST

Robustness

- Let’s consider a simpler example… linear regression

Problem: Fit a line to these datapoints



Least squares fit

- How can we fix this?

We need a better cost function…

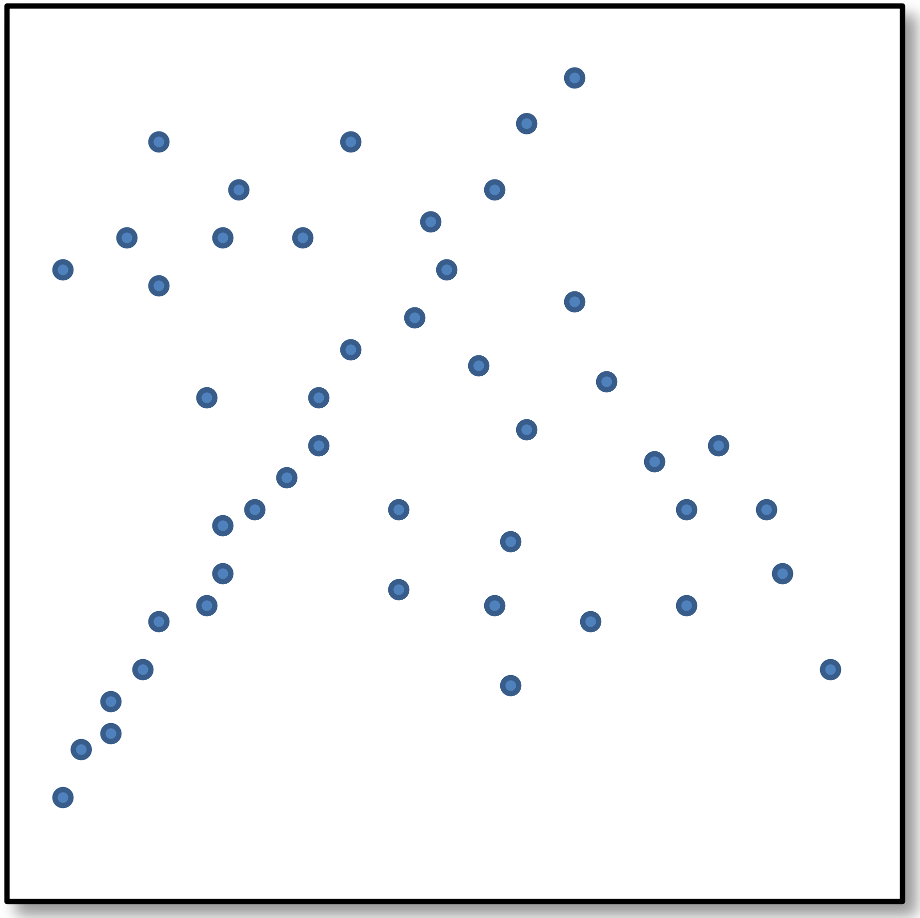
- Suggestions?

Idea

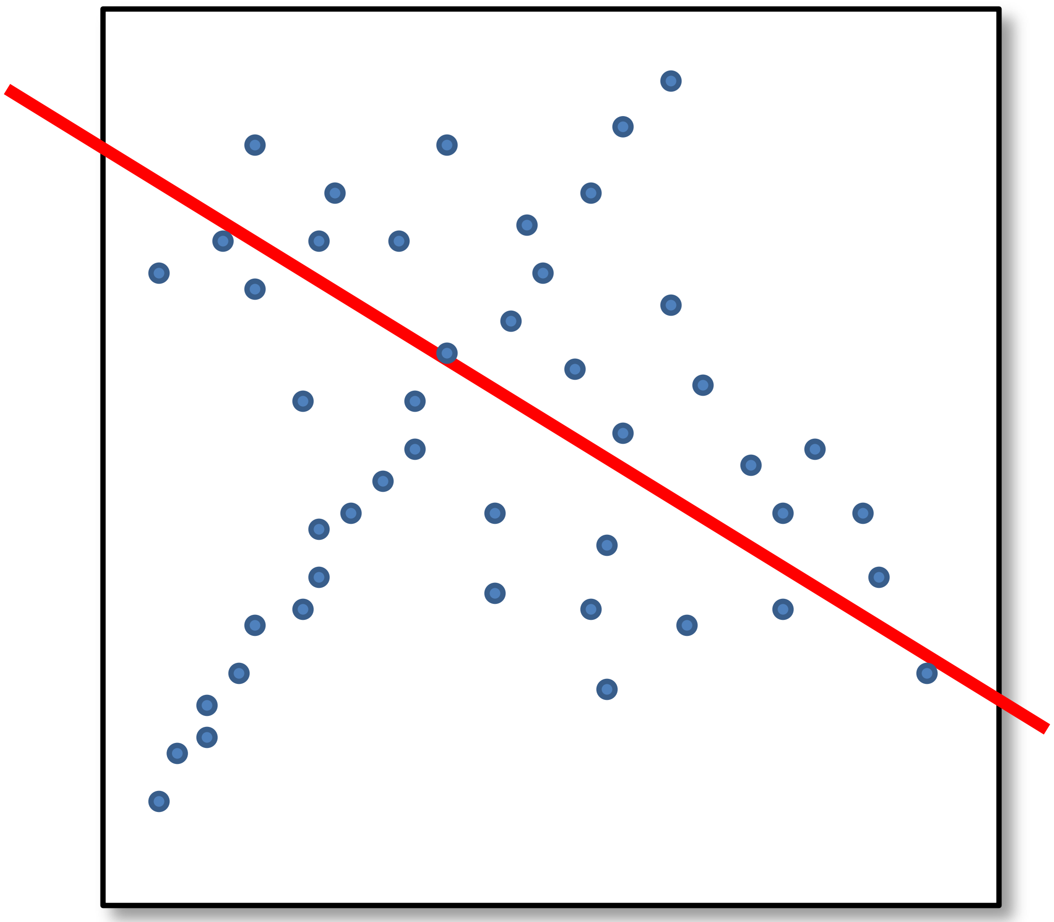
- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

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

Counting inliers

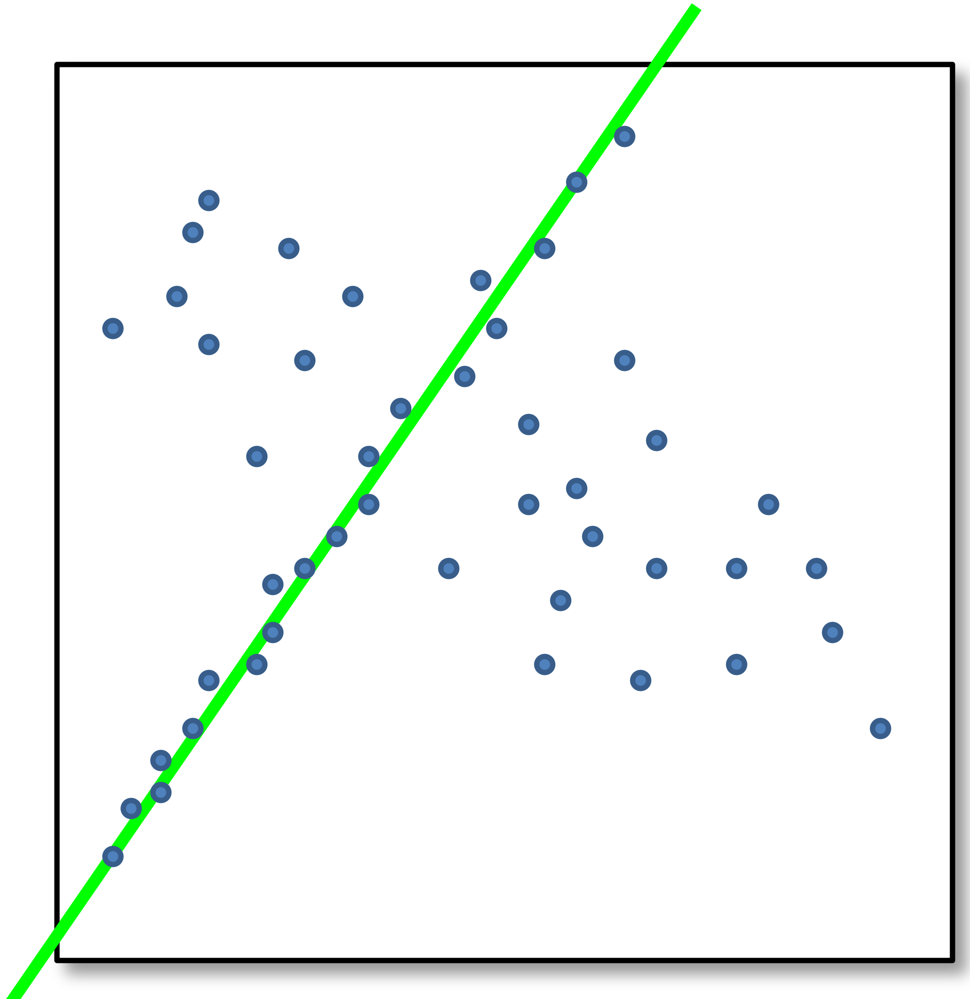


Counting inliers



Inliers: 3

Counting inliers

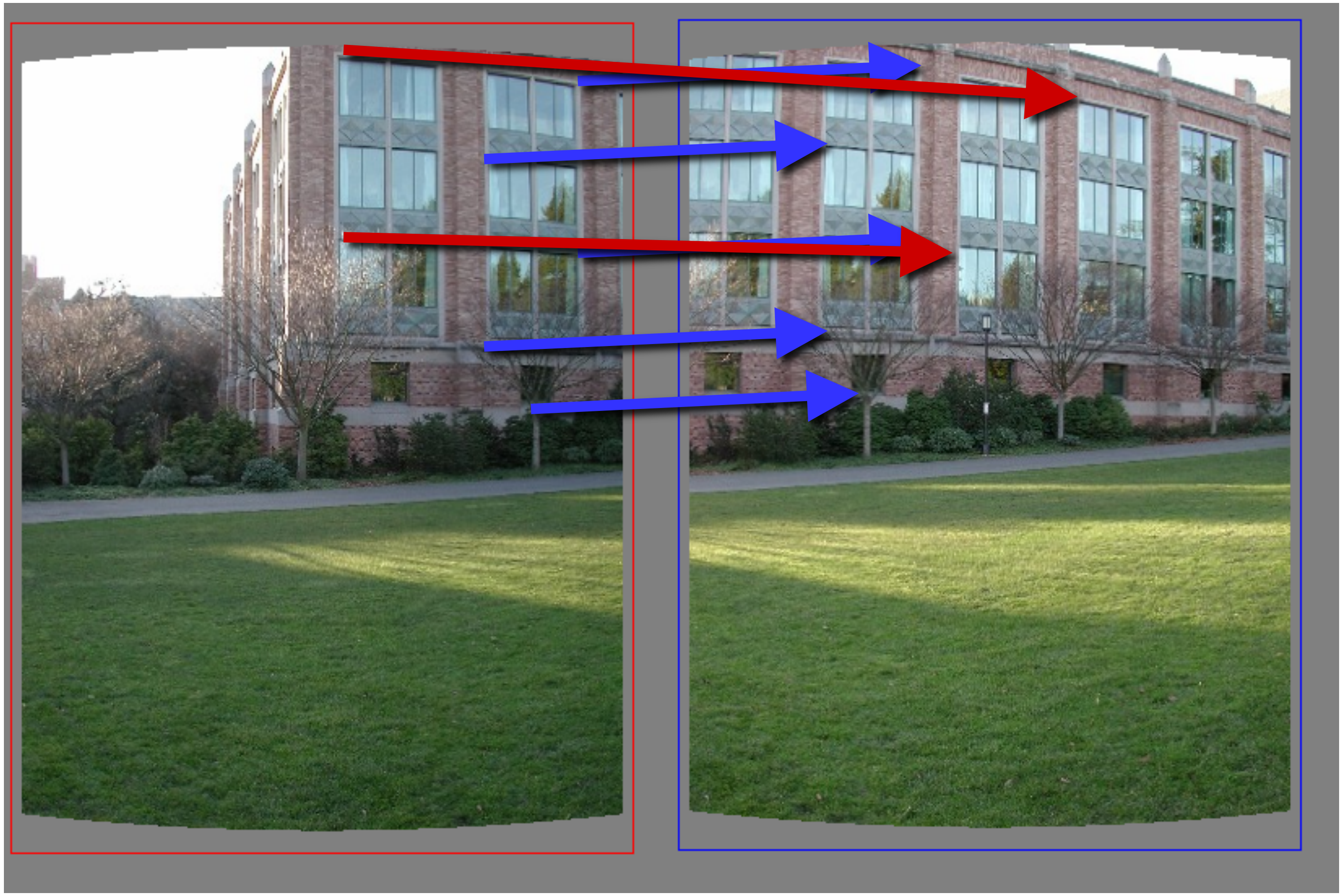


Inliers: 20

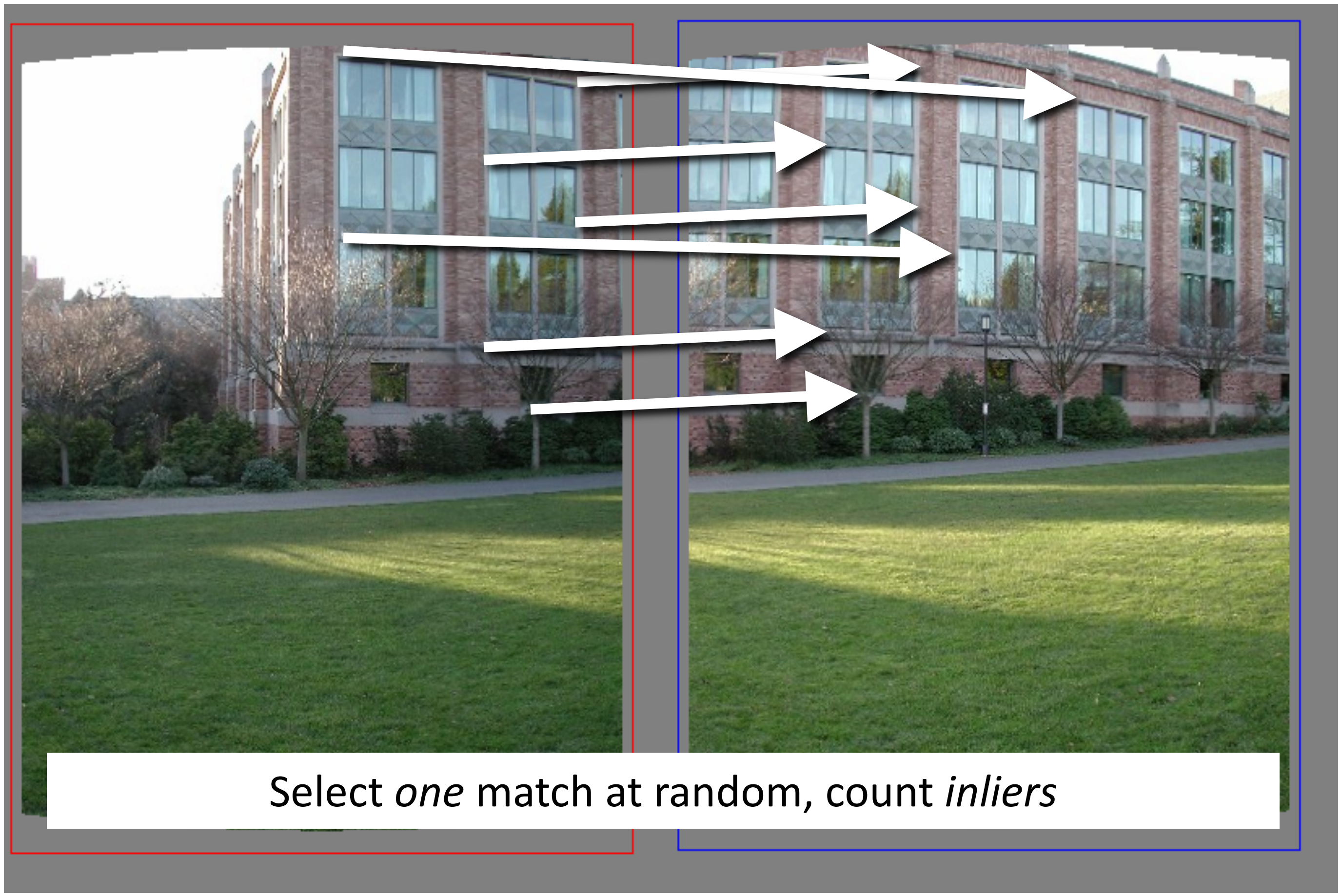
How do we find the best line?

- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines? Which one is the “best”?

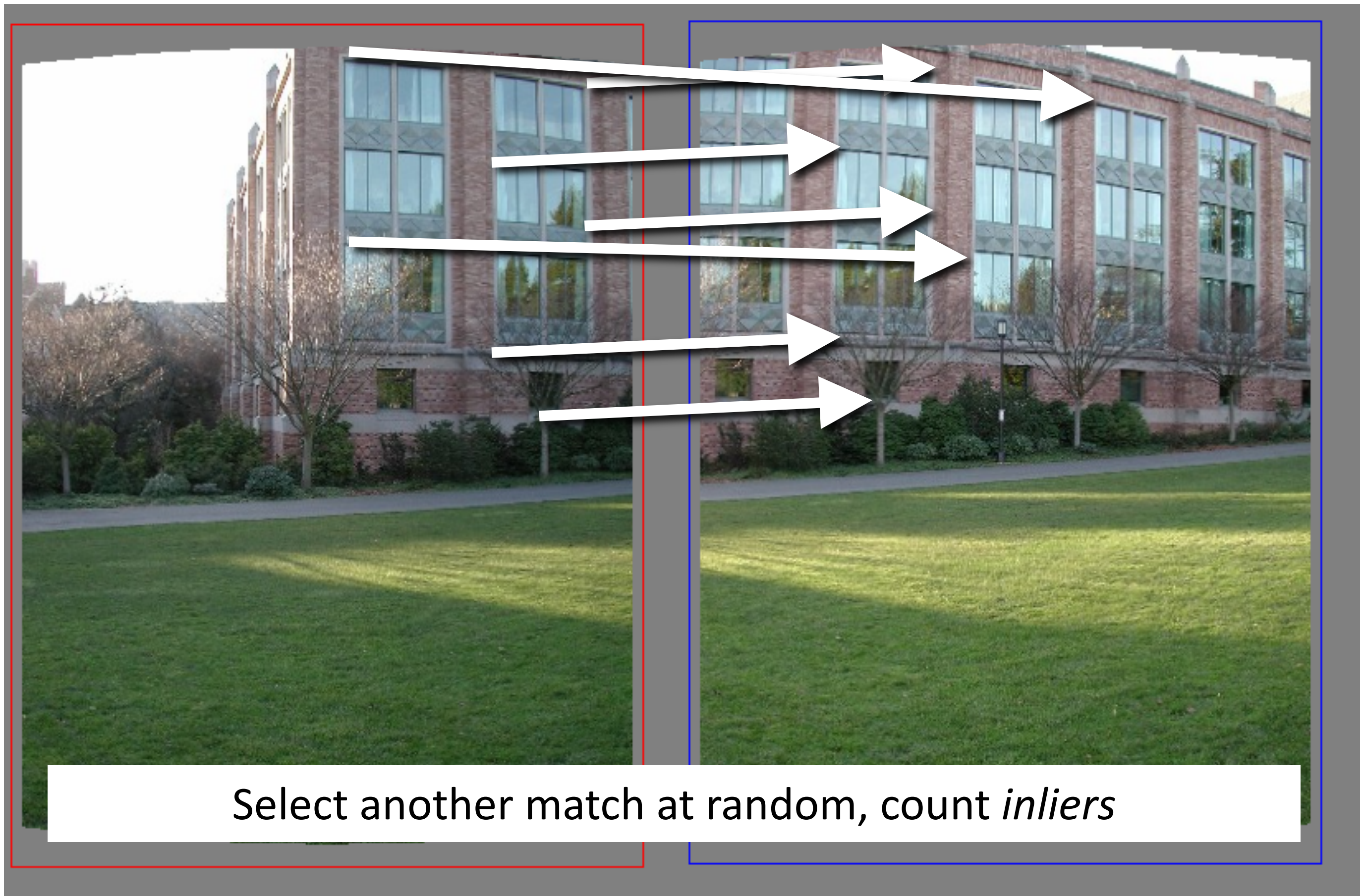
Translations



RAndom SAmple Consensus

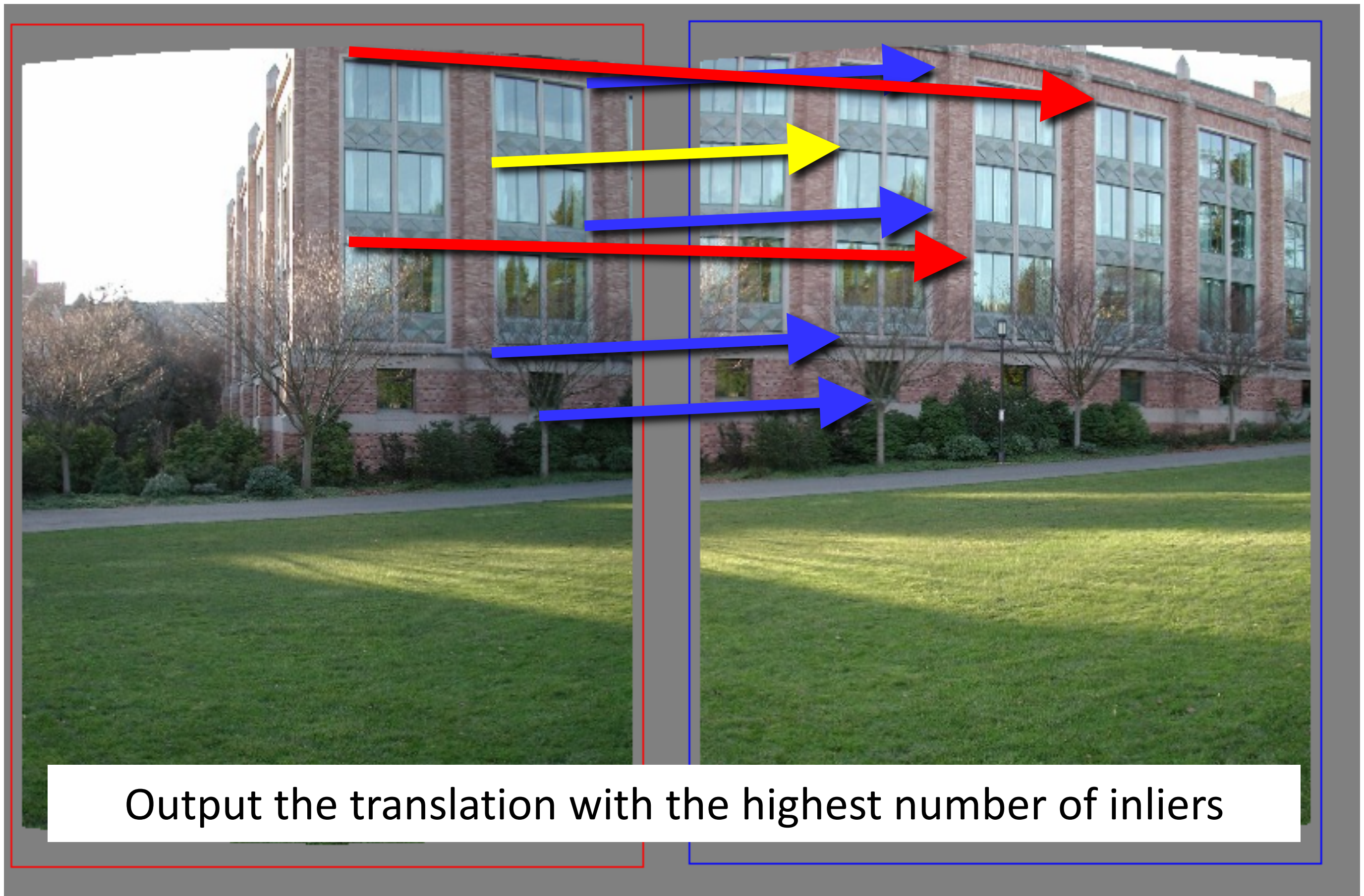


RAndom SAmple Consensus



Select another match at random, count *inliers*

RAndom SAmple Consensus



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

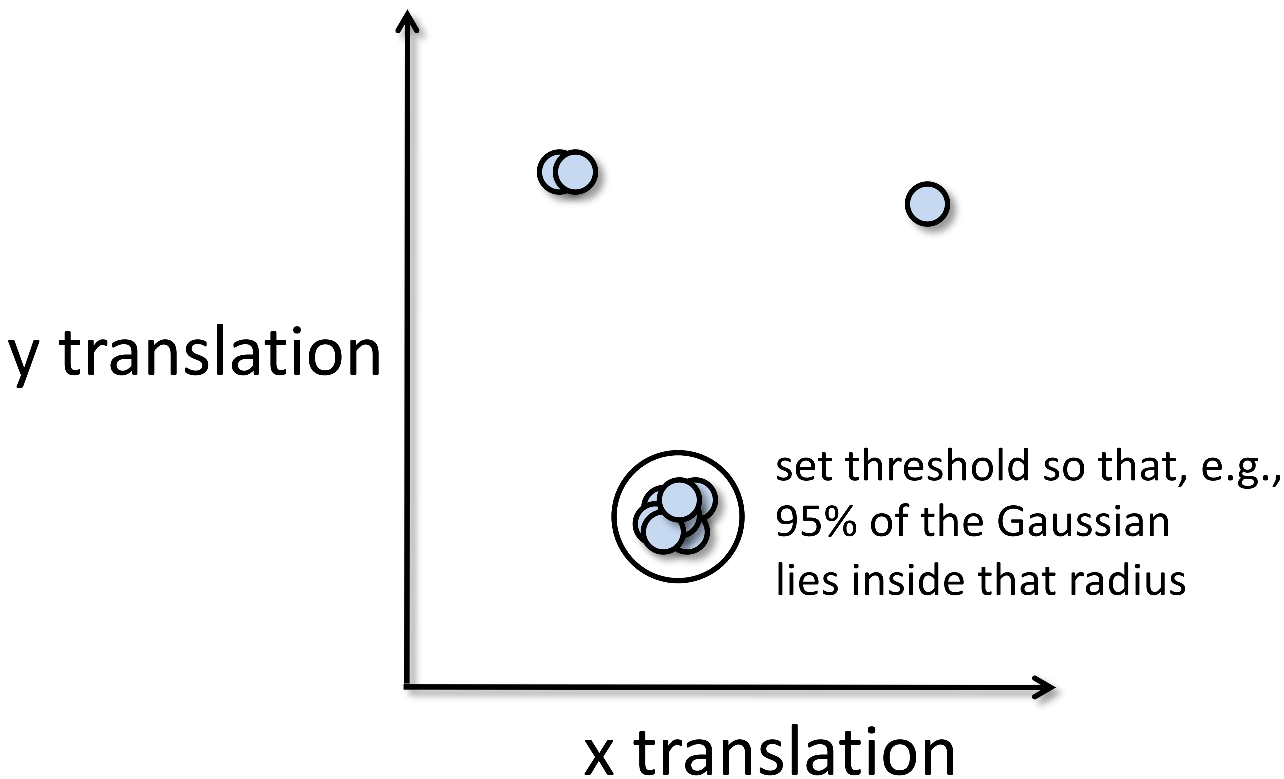
RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
  - Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
  - How many rounds do we need?

RANSAC

- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
  - Suppose there are 20% outliers, and we want to find the correct answer with 99% probability
  - How many rounds do we need?

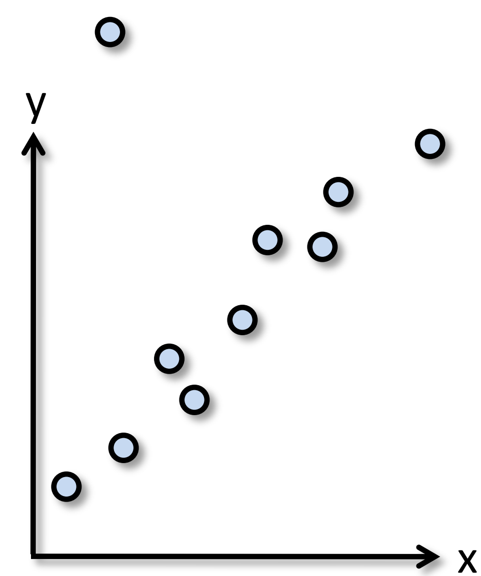
RANSAC



y translation
set threshold so that, e.g.,
95% of the Gaussian
lies inside that radius
x translation

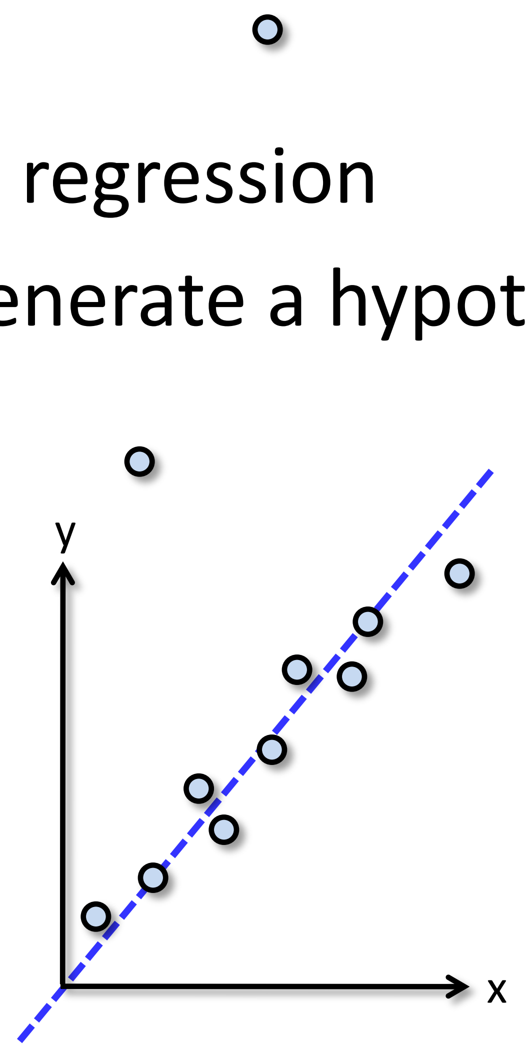
RANSAC

- Back to linear regression
- How do we generate a hypothesis?



RANSAC

- Back to linear regression
- How do we generate a hypothesis?



RANSAC

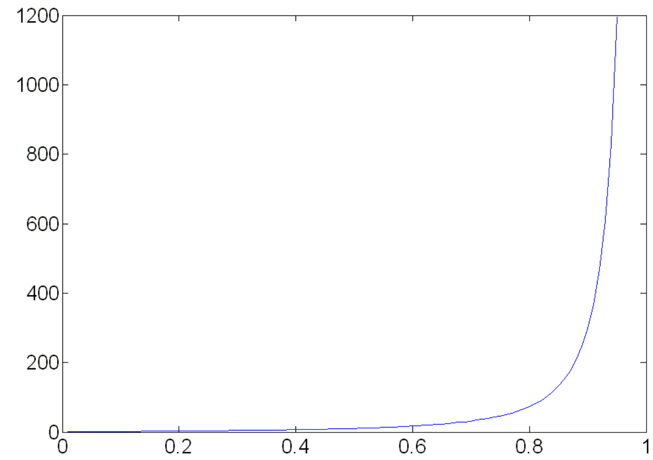
- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set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

	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

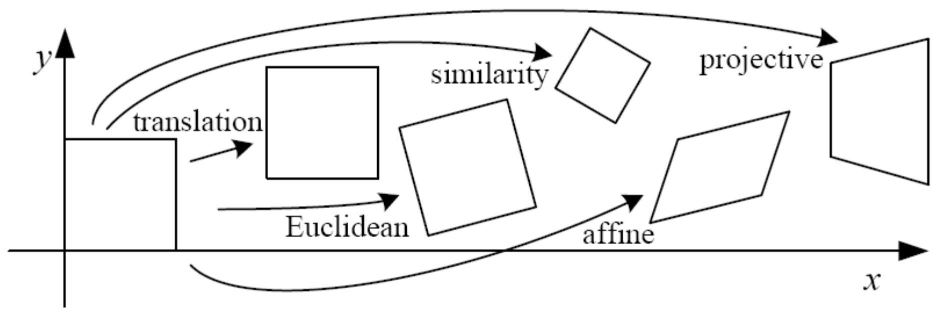
$p = 0.99$



Source: M. Pollefeys

How big is *s*?

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

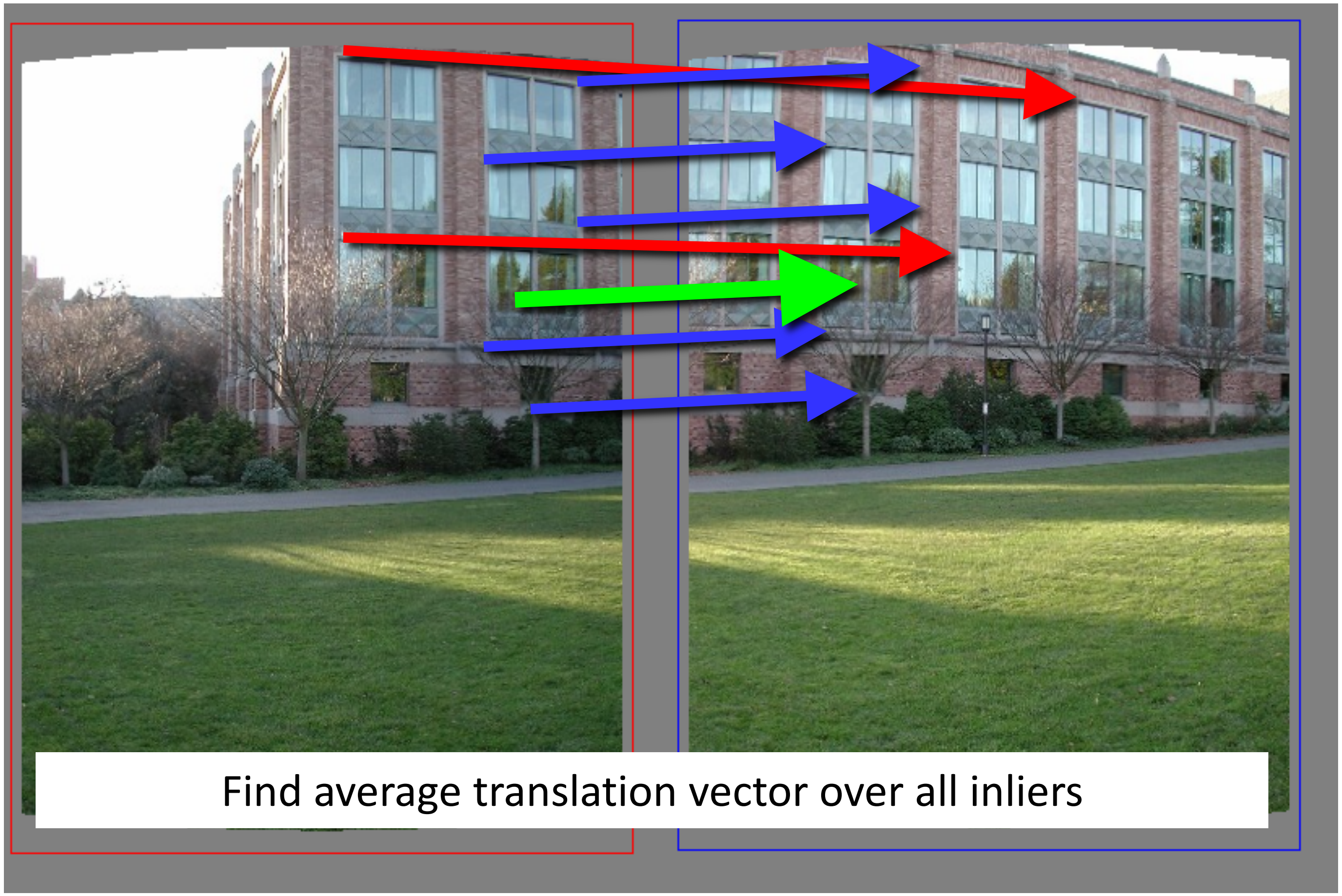


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times 3}$	4	angles $+\cdots$	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times 3}$	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

Final step: least squares fit



RANSAC

- An example of a “voting”-based fitting scheme
- Each hypothesis gets voted on by each data point, best hypothesis wins

- There are many other types of voting schemes
  - E.g., Hough transforms…

Panoramas

- Now we know how to create panoramas!

- Given two images:
  - Step 1: Detect features
  - Step 2: Match features
  - Step 3: Compute a homography using RANSAC
  - Step 4: Combine the images together (somehow)

- What if we have more than two images?