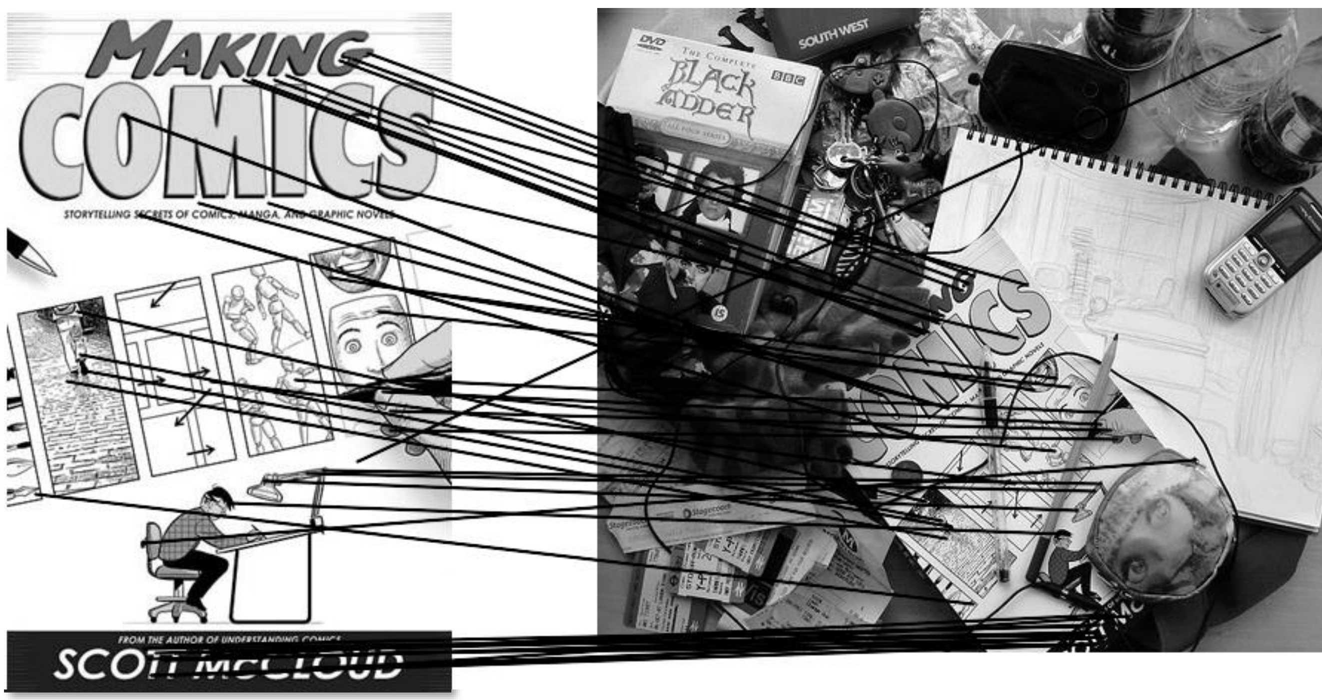
Image alignment

http://www.wired.com/gadgetlab/2010/07/camera-software-lets-you-see-into-the-past/

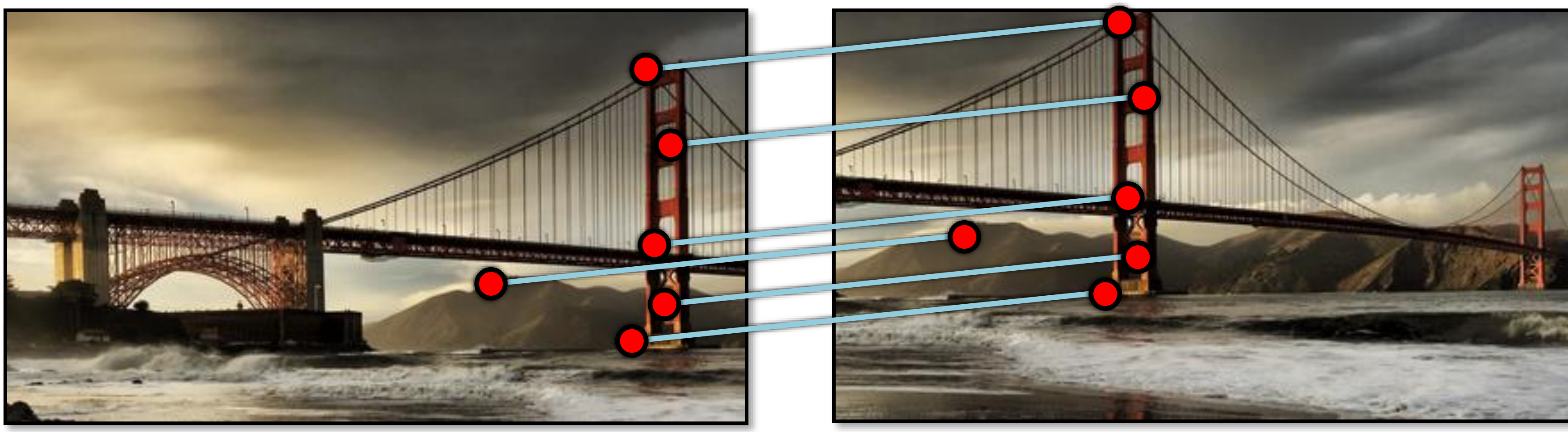
Computing transformations

- 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

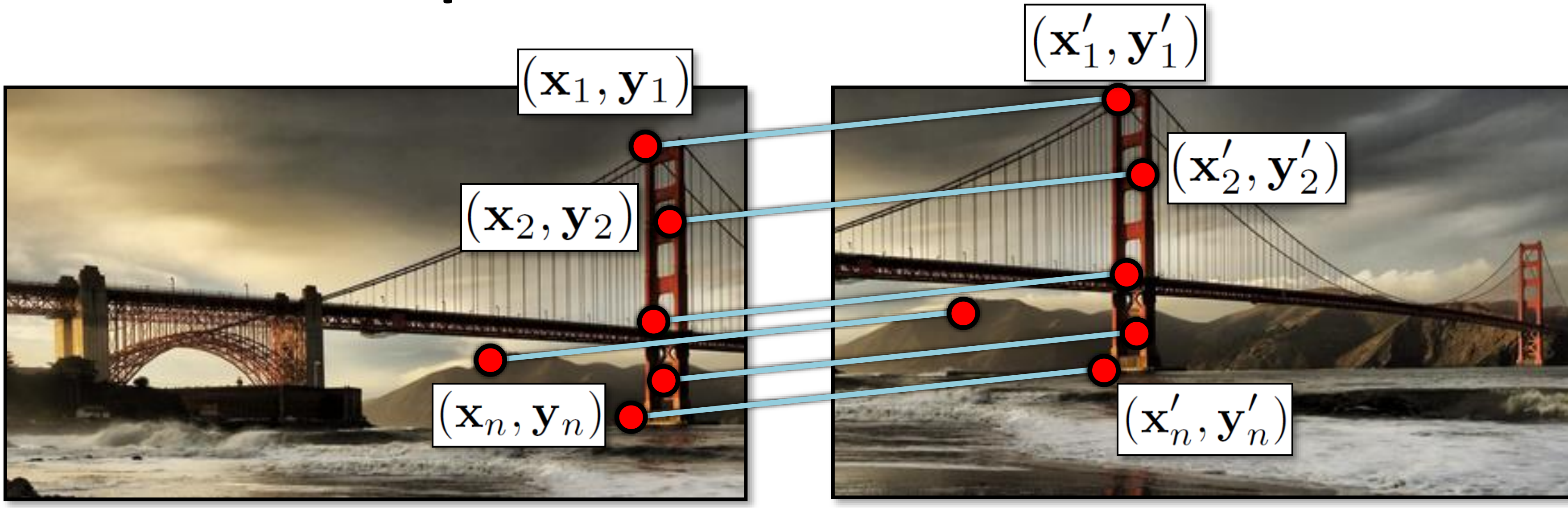
Simple case: translations



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

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

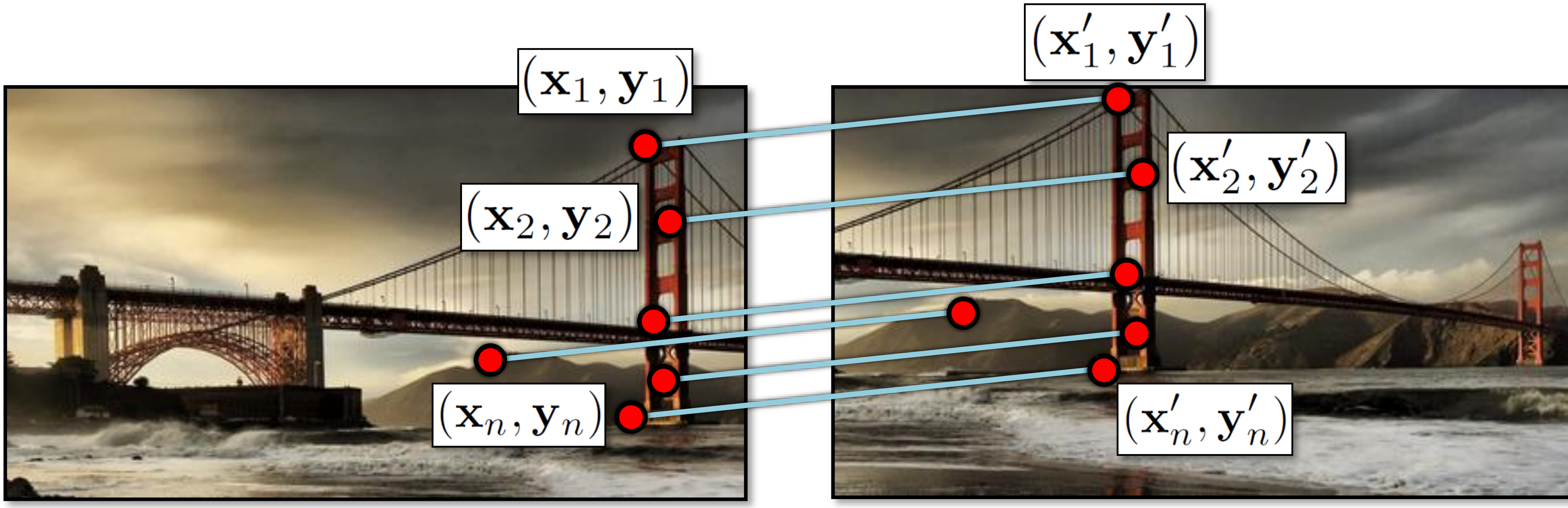
Simple case: translations



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

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

Another view

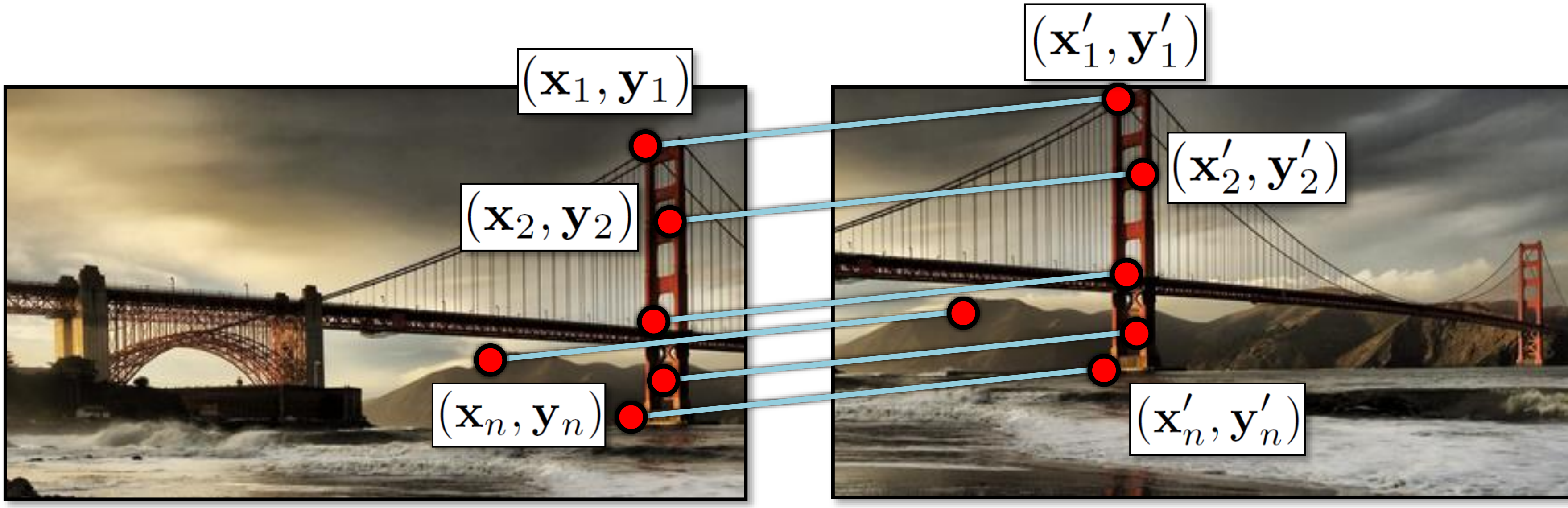


$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

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

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

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

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

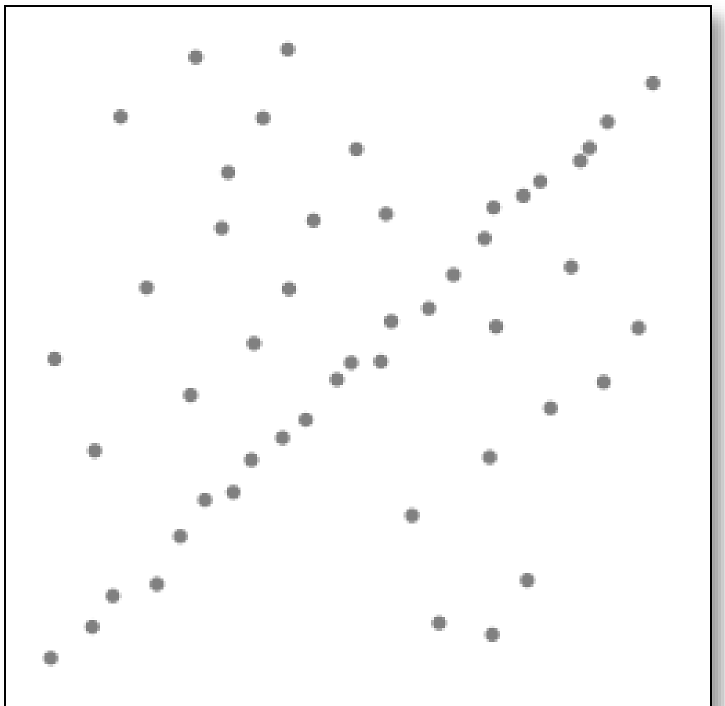
What could go wrong?

Outliers

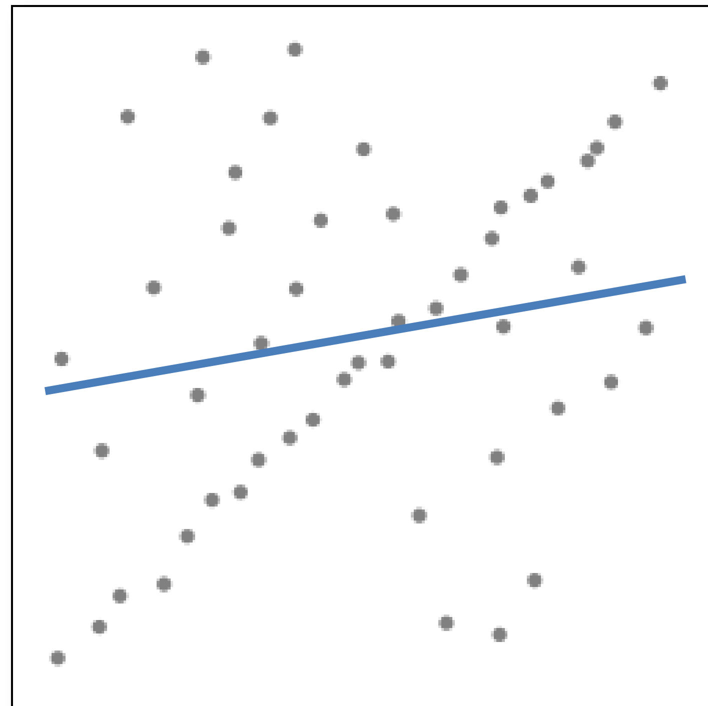

outliers
MAKING
COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
FROM THE AUTHOR OF UNDERSTANDING COMICS
SCOTT McCLOUD
DVD
THE COMPLETE
BLACK ADDER
BBC
SOUTH WEST
inliers

Robustness

- Let's consider a simpler example… linear regression



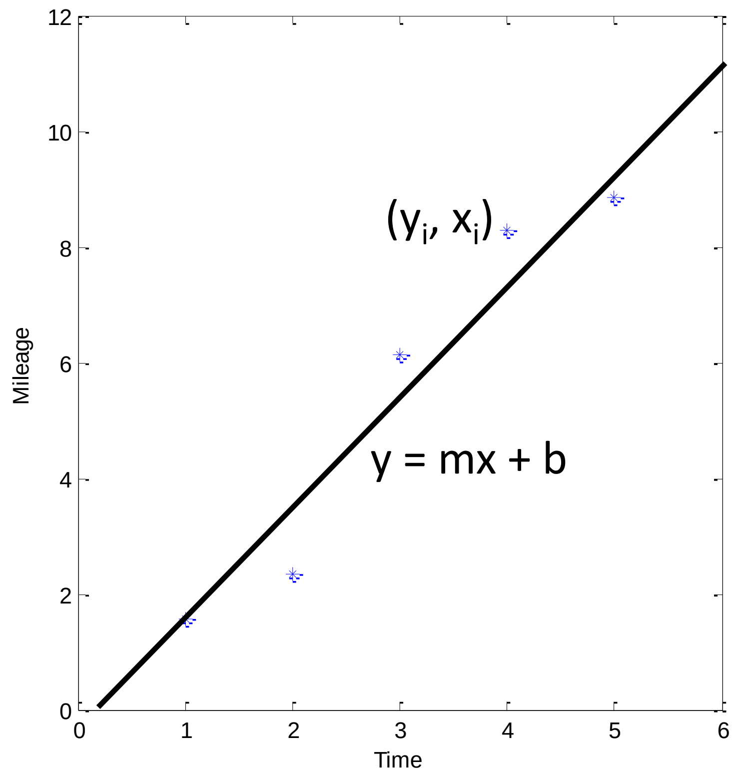
Problem: Fit a line to these datapoints



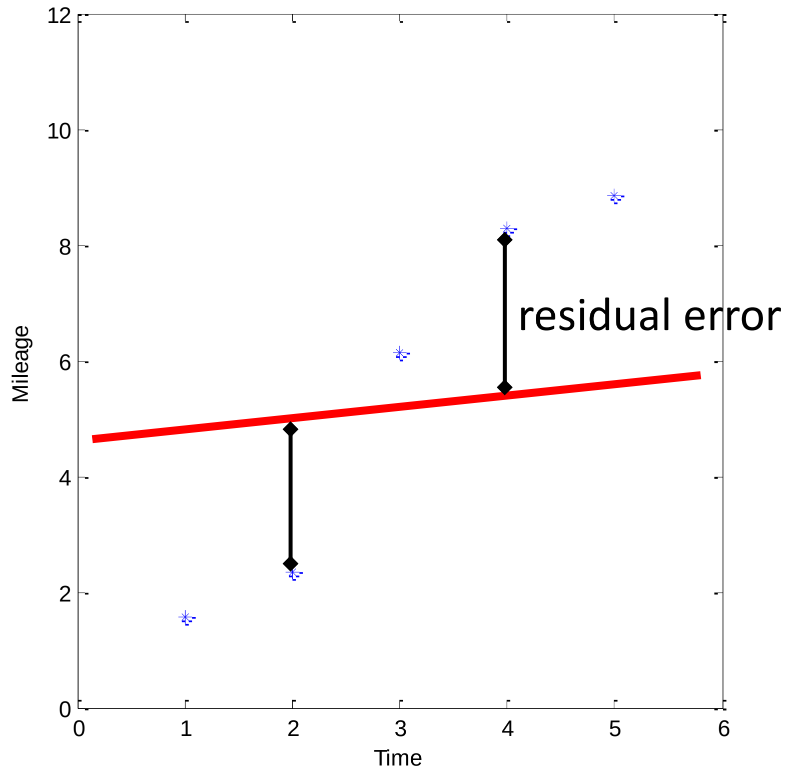
Least squares fit

- How can we fix this?

Least squares: linear regression



Linear regression



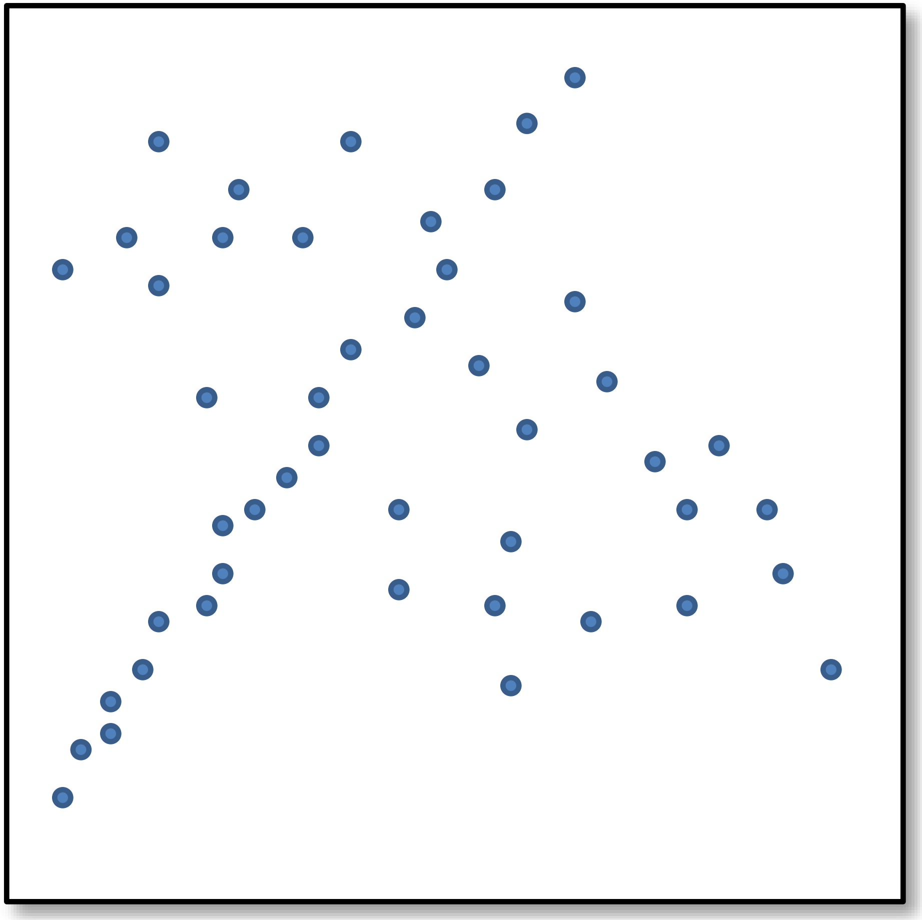
$$\mathrm{Cost}(m,b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2$$

Idea

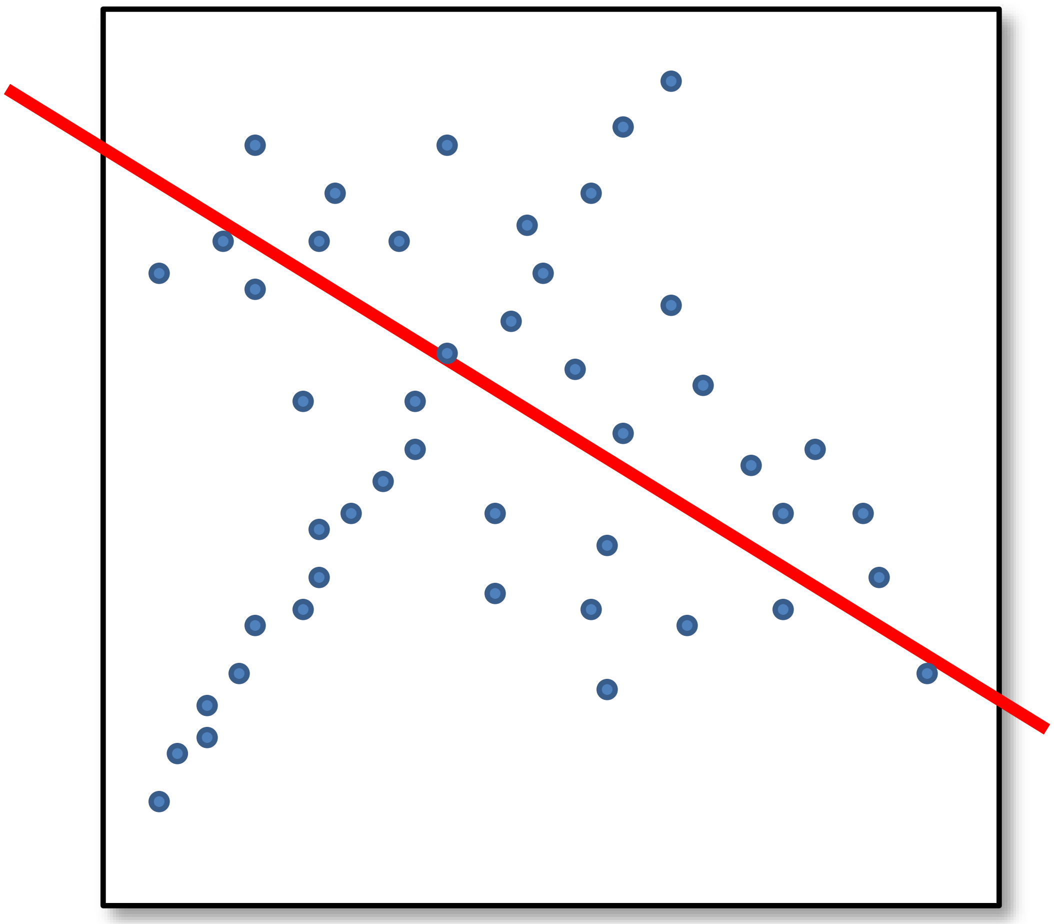
- Given a hypothesized line
- Count the number of points that “agree” with the line
  – “Agree” = within a small distance of the line
  – I.e., the **inliers** to that line

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

Counting inliers

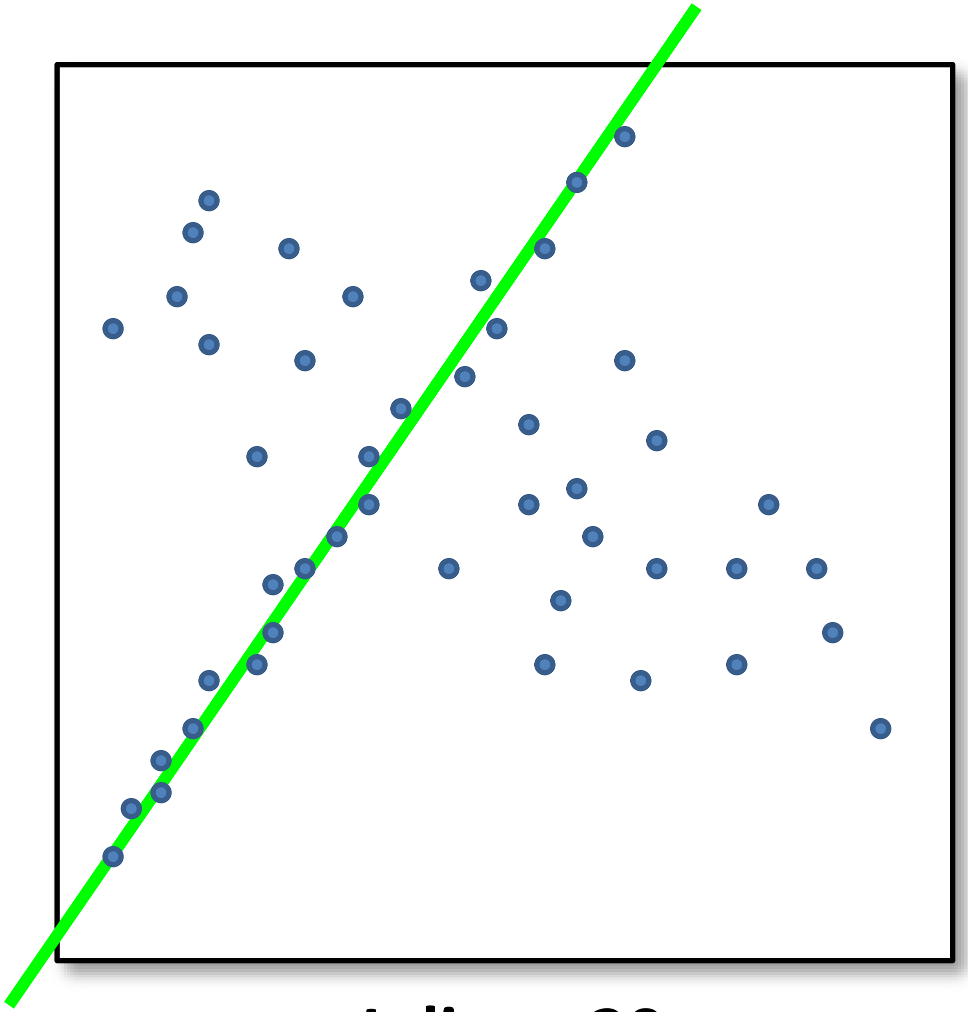


Counting inliers



Inliers: 3

Counting inliers

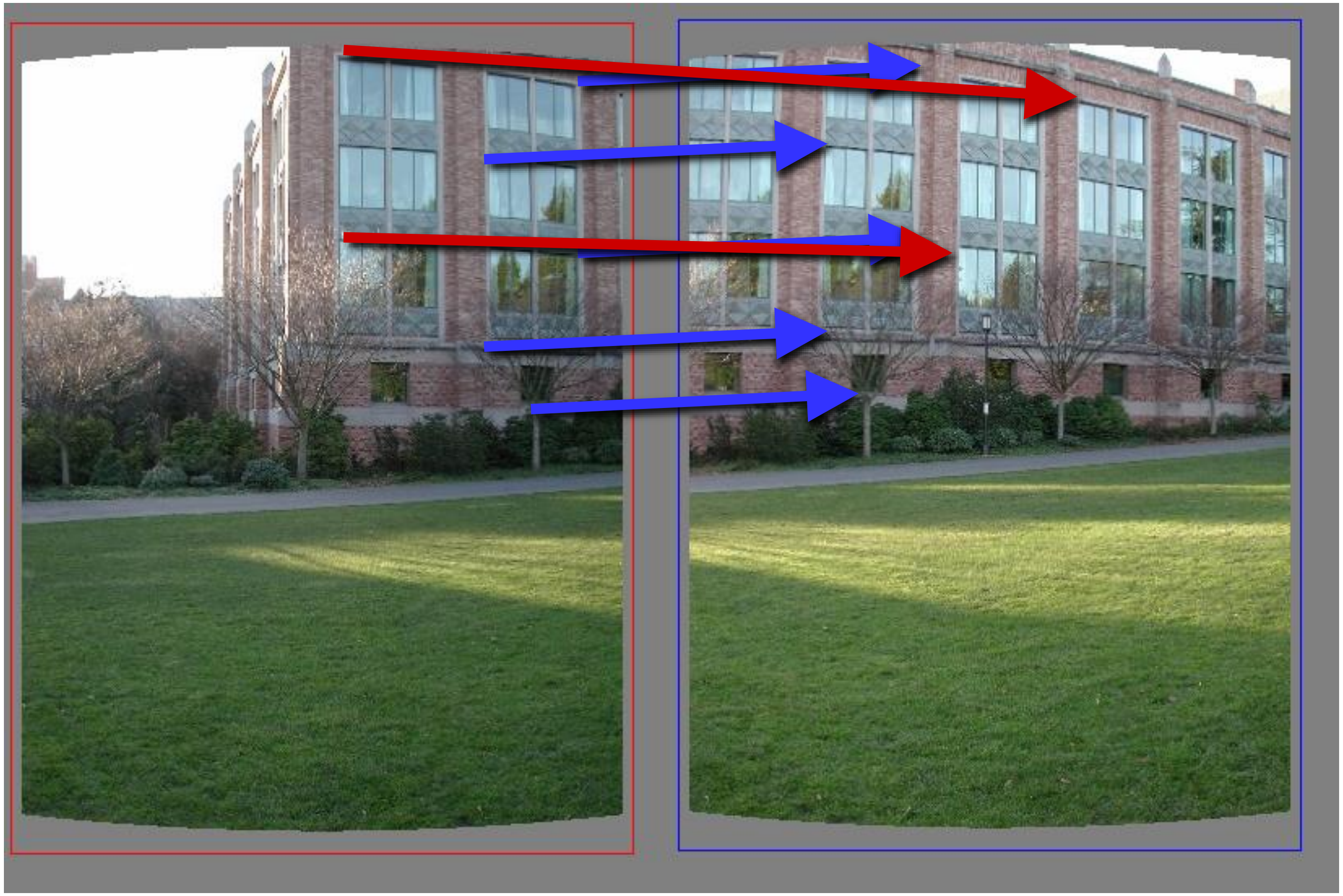


Inliers: 20

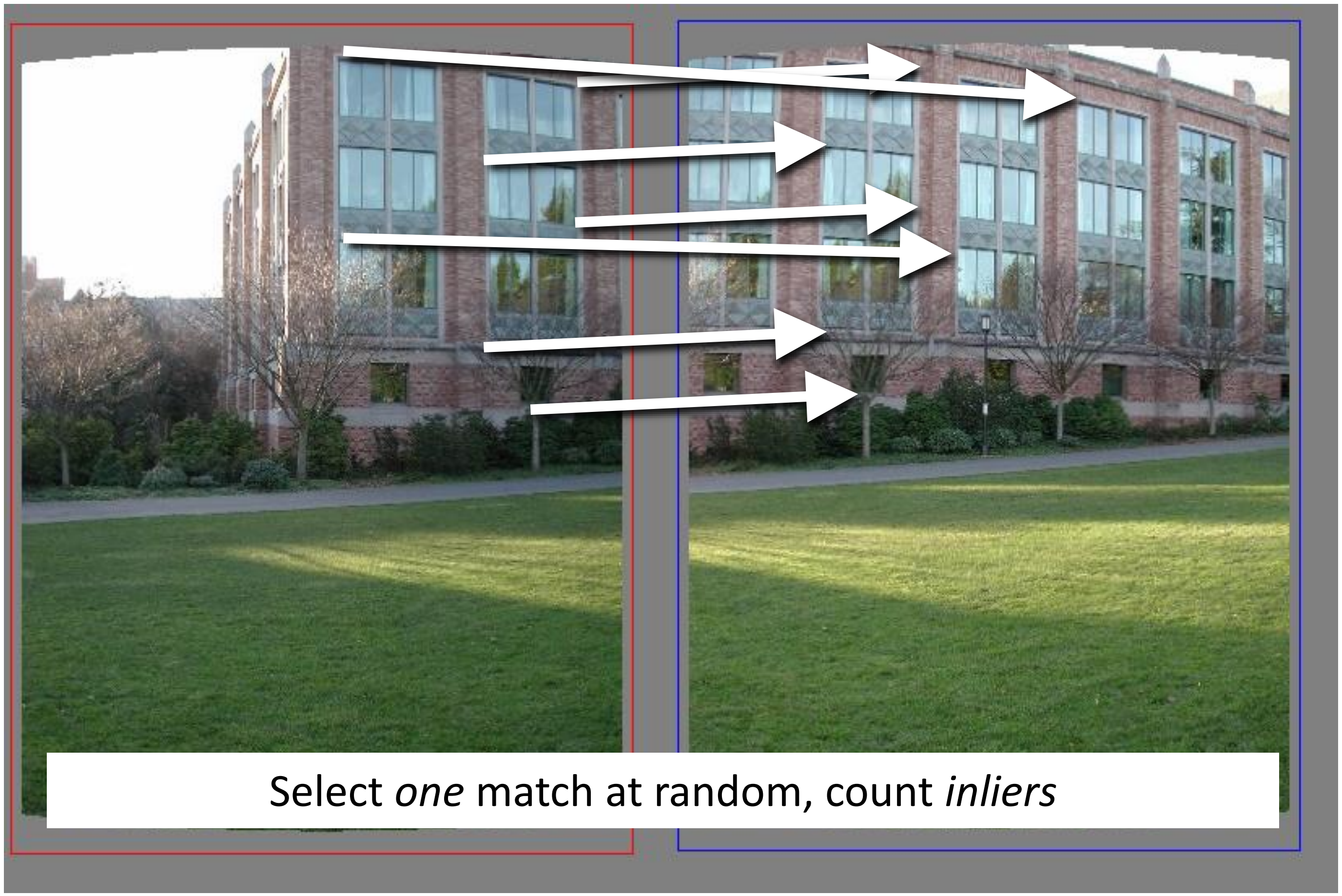
How do we find the best line?

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

Translations

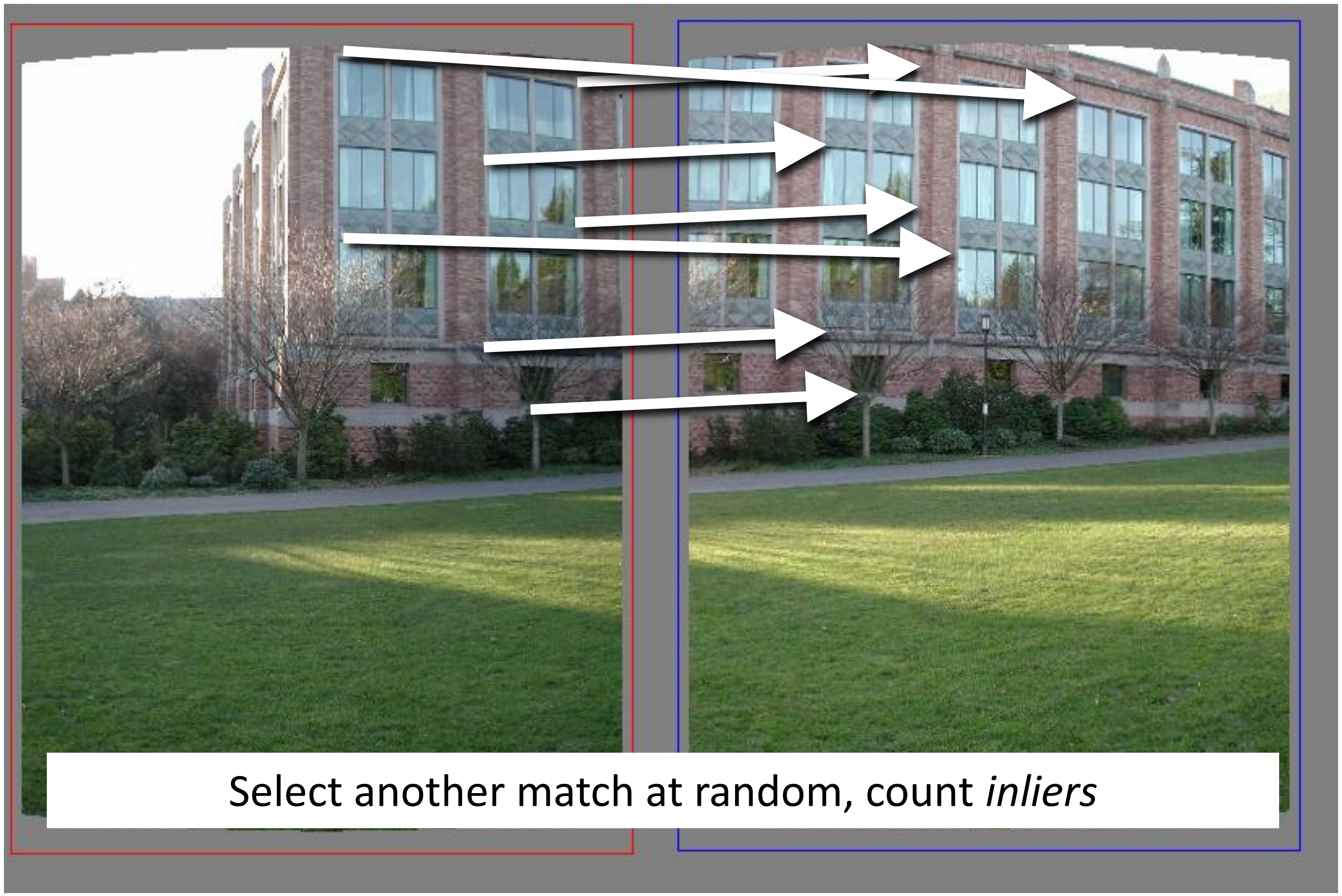


RAndom SAmple Consensus

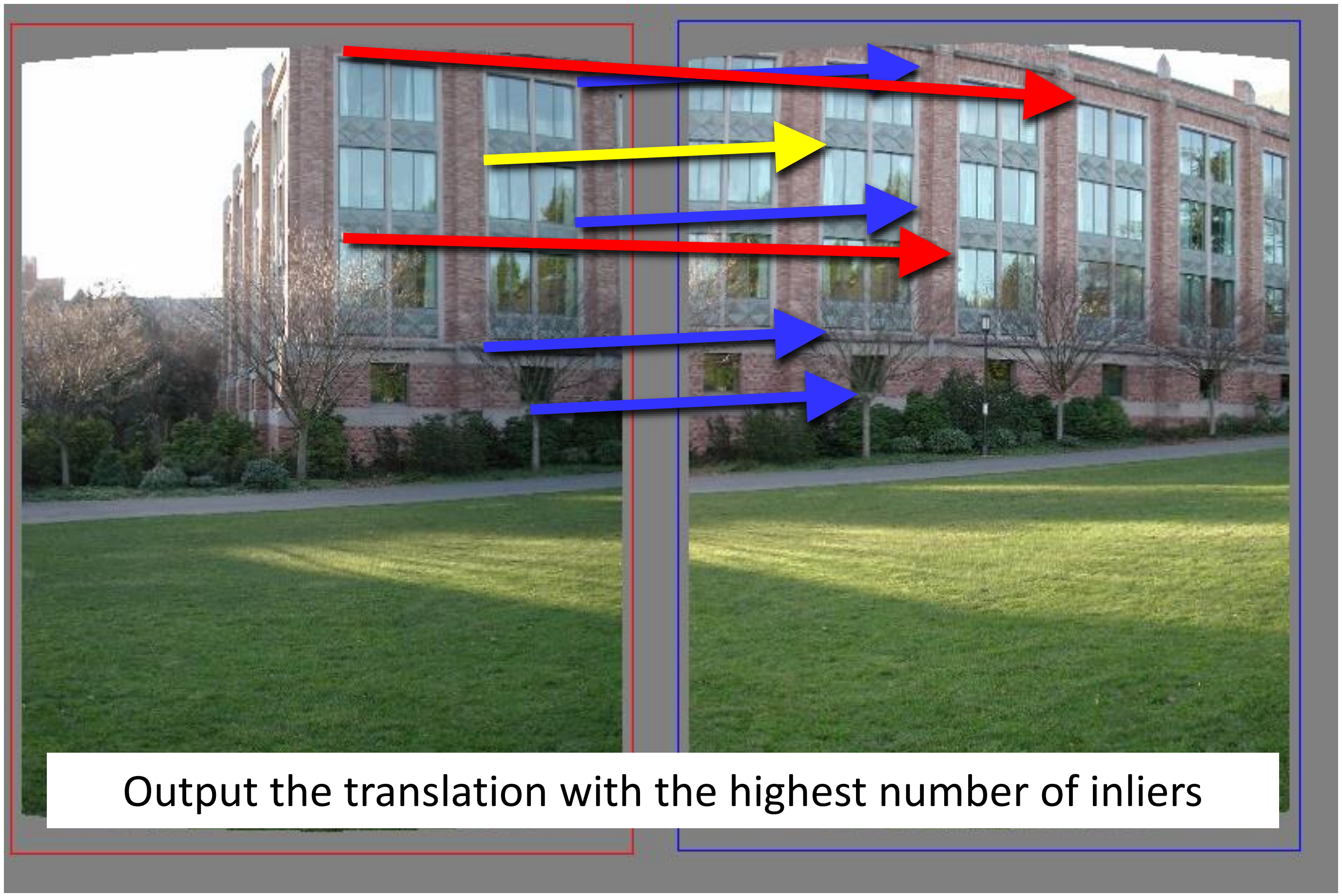


Select *one* match at random, count *inliers*

RAndom SAmple Consensus



RAndom SAmple Consensus



Output the translation with the highest number of inliers

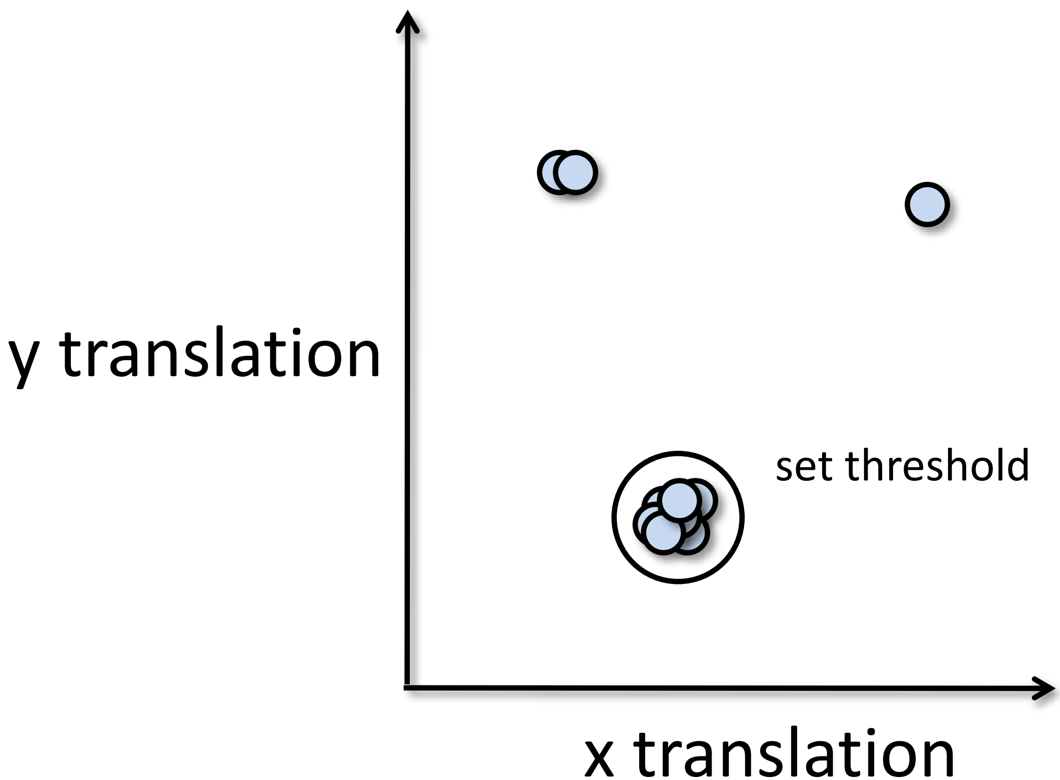
RANSAC

- 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

  - “All good matches are alike; every bad match is bad in its own way.”

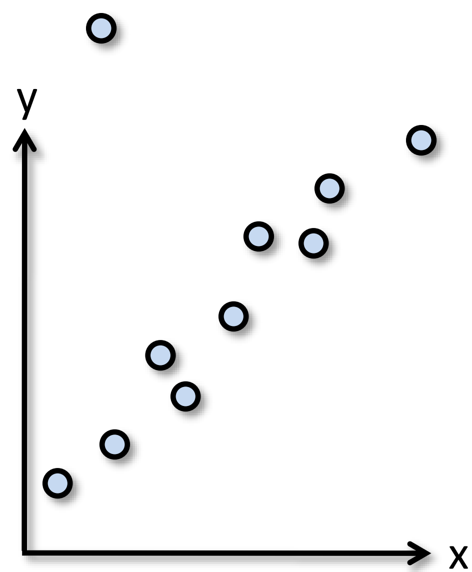
– Tolstoy via Alyosha Efros

RANSAC



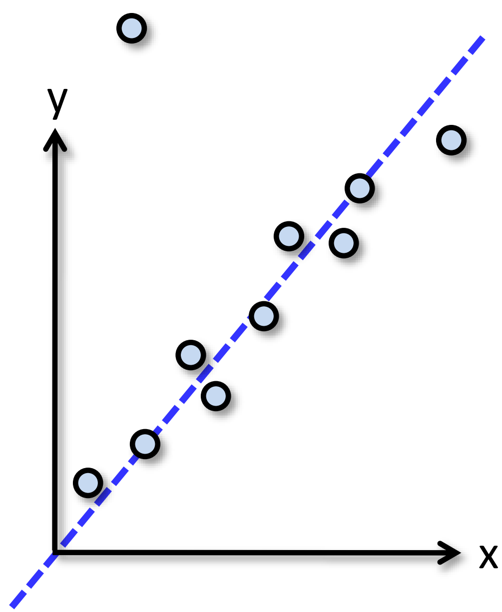
RANSAC

- 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

Final step: least squares fit