CS 4610/5335 – Lecture 19
Point cloud processing

Lawson L.S. Wong
Northeastern University
4/6/22

Material adapted from:
1. Robert Platt, CS 4610/5335
2. Peter Corke, Robotics, Vision and Control
3. Wolfram Burgard, U. Freiburg Mobile Robotics Course
4. Kavita Bala, Cornell CS 4670/5760

Announcements

Ex5 out, due 4/15 (Fri) – the final one !

Final project report due in 1 month (May 6) !!!

May try to adjust office hours (this Friday / future weeks)
(working on it)

Upcoming schedule:

4/11	(Mon)	Project check-ins
4/13	(Wed)	Learning for perception and action
4/18	*(Mon)*	*Patriots' Day (no class)*
4/20	(Wed)	More on learning...

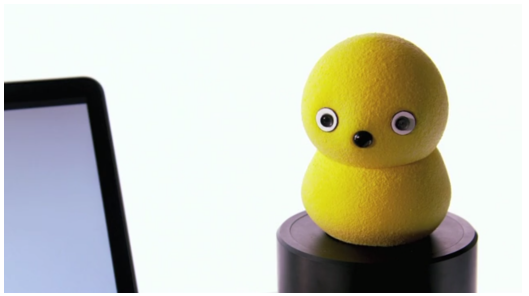
…

Robotics talk of interest!

Sam Spaulding
Ph.D. Candidate
MIT Media Lab

Lifelong Personalization for
Social Language/Literacy Companions:
Interactive Student Modeling
Across Tasks and Over Time

Thursday, April 7
11:30 AM – 12:30 PM
366 West Village H
also on Zoom: 981 7931 0990 (317281)



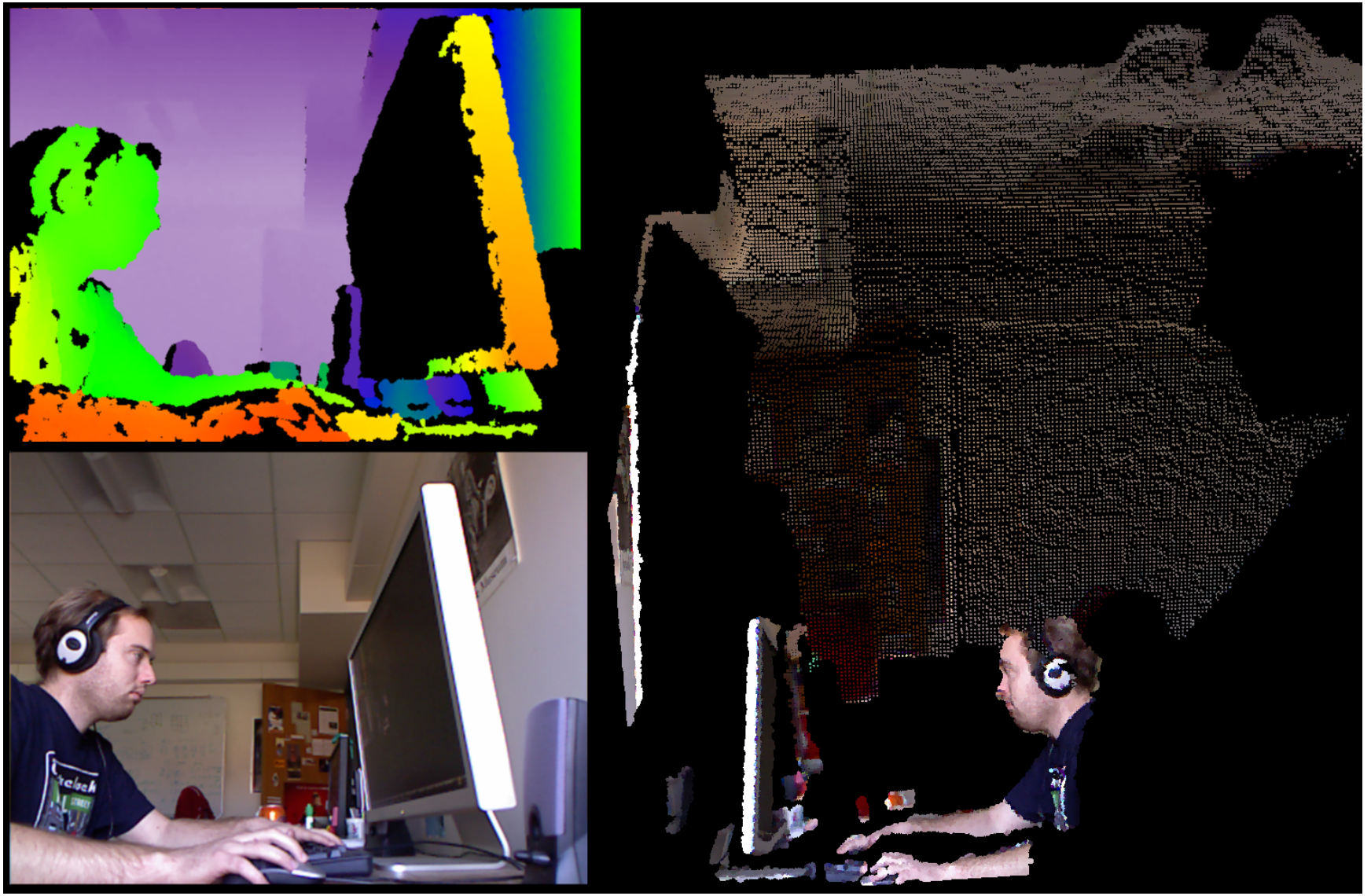
Outline

Aligning point clouds
- Iterative closest point

Extracting primitive shapes
- Surface normals
- Planes
- RANSAC
- More shapes (time permitting)

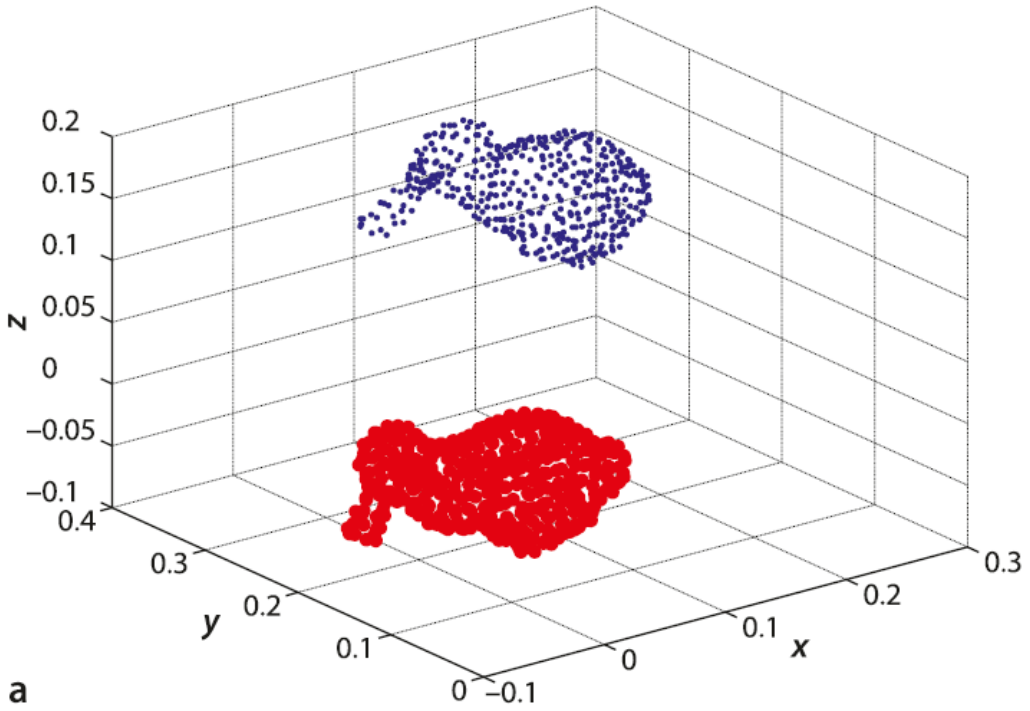
Advanced perception (time permitting)

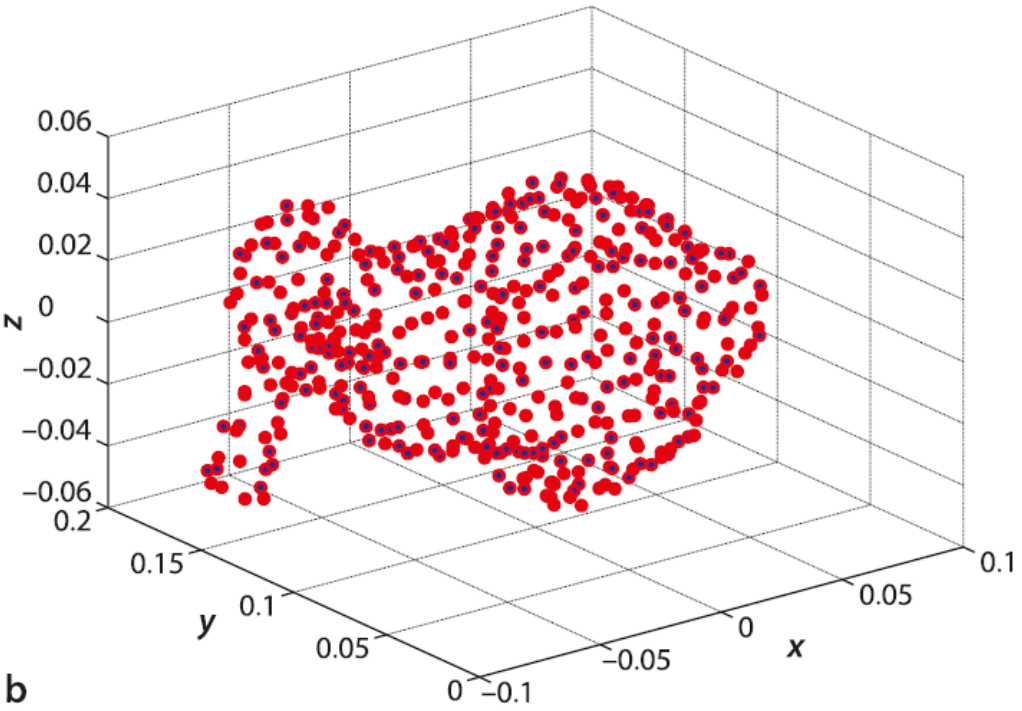
RGB-D image



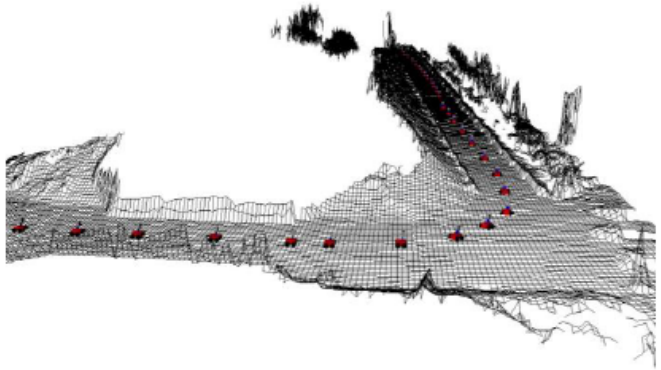
Aligning point clouds

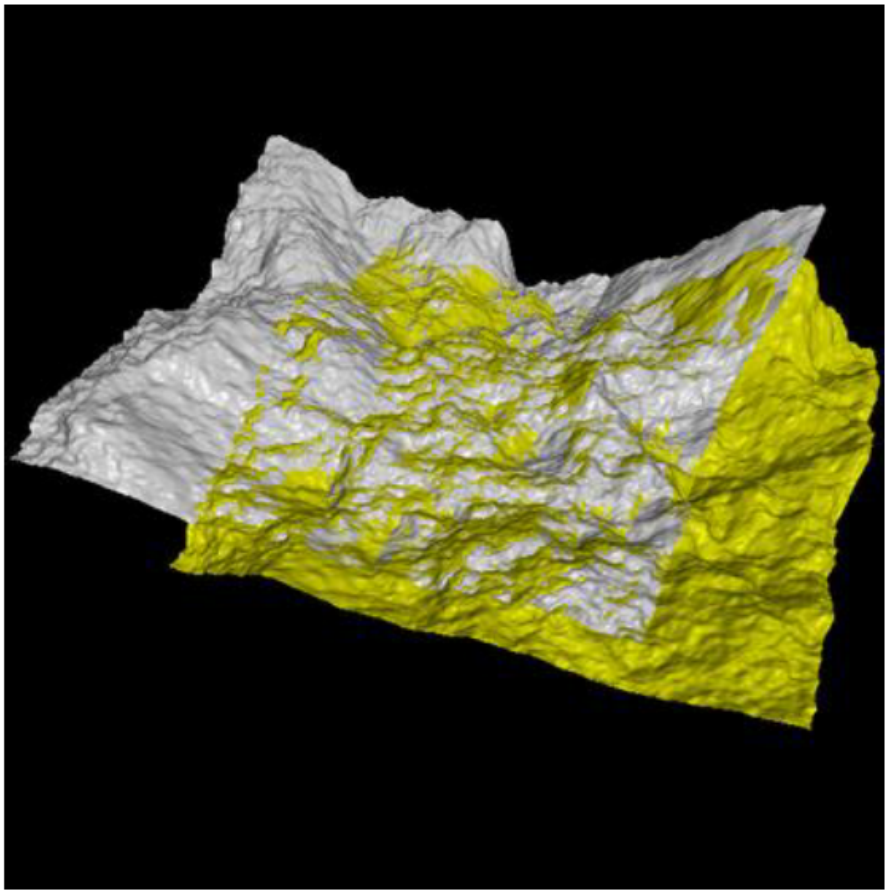
Fig. 14.42. Iterated closest point (ICP) matching of two point clouds: model (*red*) and data (*blue*) **a** before registration, **b** after registration; observed data points have been transformed to the model coordinate frame using the inverse of the identified transformation (Stanford bunny model courtesy Stanford Graphics Laboratory)





Iterative closest point (ICP)





Goal: Find local transformation to align points

Iterative closest point (ICP)

- ICP is a powerful algorithm for calculating the displacement between scans
- The major problem is to determine the correct data associations

Iterative closest point (ICP)

- ICP is a powerful algorithm for calculating the displacement between scans
- The major problem is to determine the correct data associations

- Given the correct data associations, the transformation can be computed efficiently using SVD

Iterative closest point (ICP)

- Given two corresponding point sets:

$$X = \{x_1, \ldots, x_{N_x}\}$$

$$P = \{p_1, \ldots, p_{N_p}\}$$

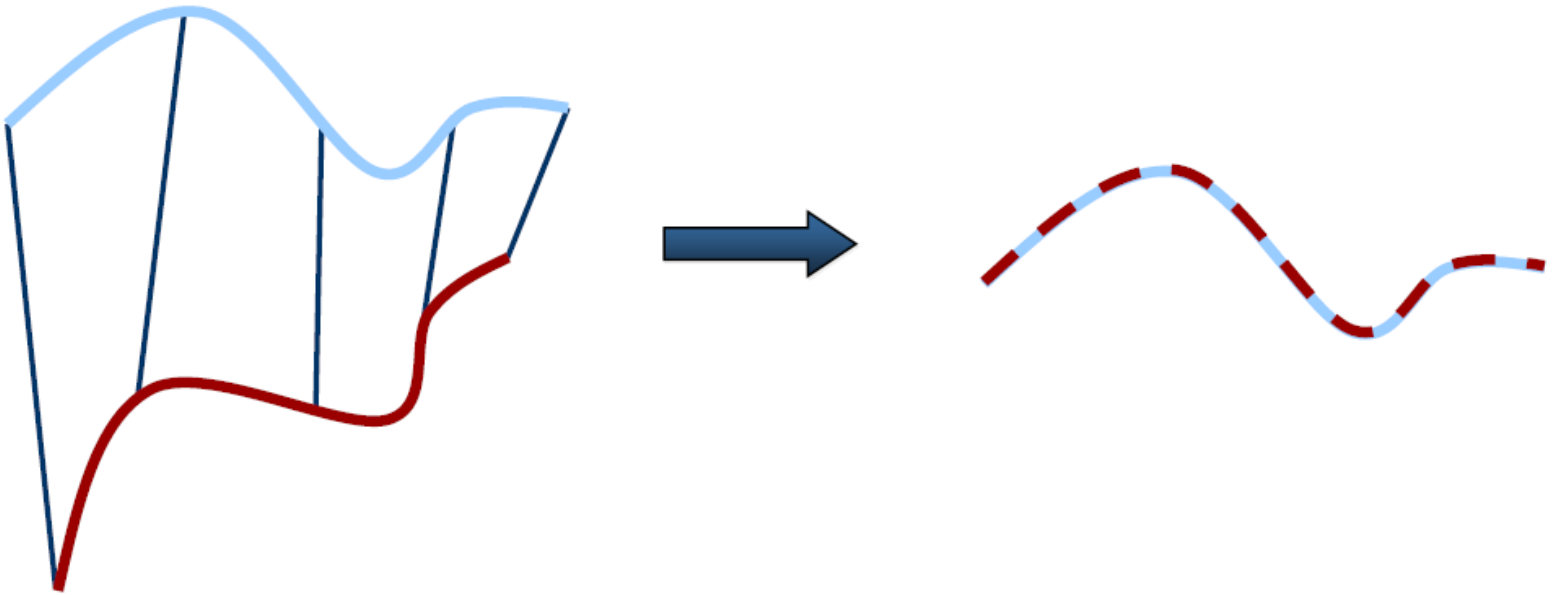
- Wanted: Translation *t* and rotation *R* that minimize the sum of the squared errors:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2$$

Here, x_i and p_i are corresponding points

ICP: Key idea

- If the correct correspondences are known, the correct relative rotation/translation can be calculated in **closed form**



ICP step 1: Center the two point clouds

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \quad \text{and} \quad \mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

are the centers of mass of the two point sets

Idea:

- Subtract the corresponding center of mass from every point in the two point sets before calculating the transformation
- The resulting point sets are:

$$X' = \{x_i - \mu_x\} = \{x'_i\}$$
$$P' = \{p_i - \mu_p\} = \{p'_i\}$$

and

ICP step 2: Use SVD to get minimizing *t* and *R*

Let $W = \sum_{i=1}^{N_p} x'_i p'^T_i$

denote the singular value decomposition (SVD) of W by:

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

where $U, V \in \mathbb{R}^{3\times3}$ are unitary, and

$\sigma_1 \geq \sigma_2 \geq \sigma_3$ are the singular values of W

ICP step 2: Use SVD to get minimizing *t* and *R*

Theorem (without proof):

If rank(W) = 3, the optimal solution of E(R,t) is unique and is given by:

$$R = UV^T$$
$$t = \mu_x - R\mu_p$$

ICP step 2: Use SVD to get minimizing *t* and *R*

Theorem (without proof):

If rank(*W*) = 3, the optimal solution of E(*R*,*t*) is unique and is given by:

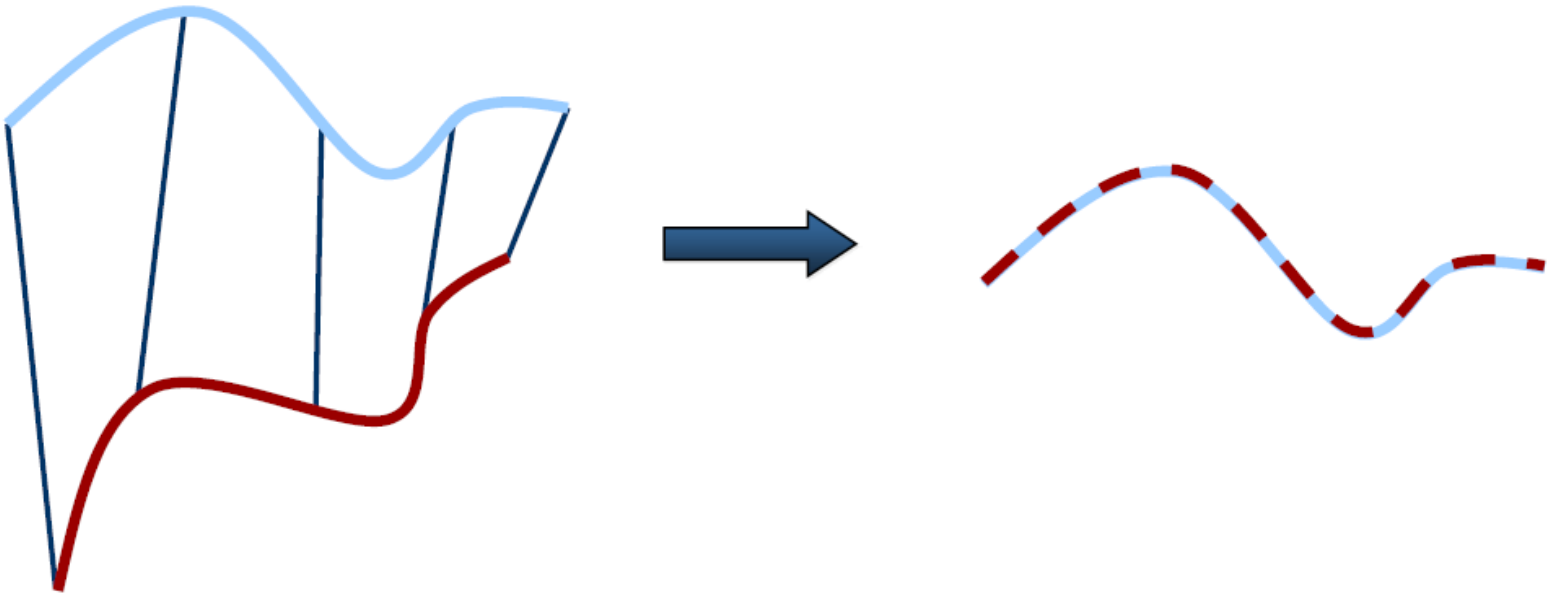
$$R = UV^T$$
$$t = \mu_x - R\mu_p$$

The minimal value of error function at (*R*,*t*) is:

$$E(R,t) = \sum_{i=1}^{N_p} (||x_i'||^2 + ||y_i'||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

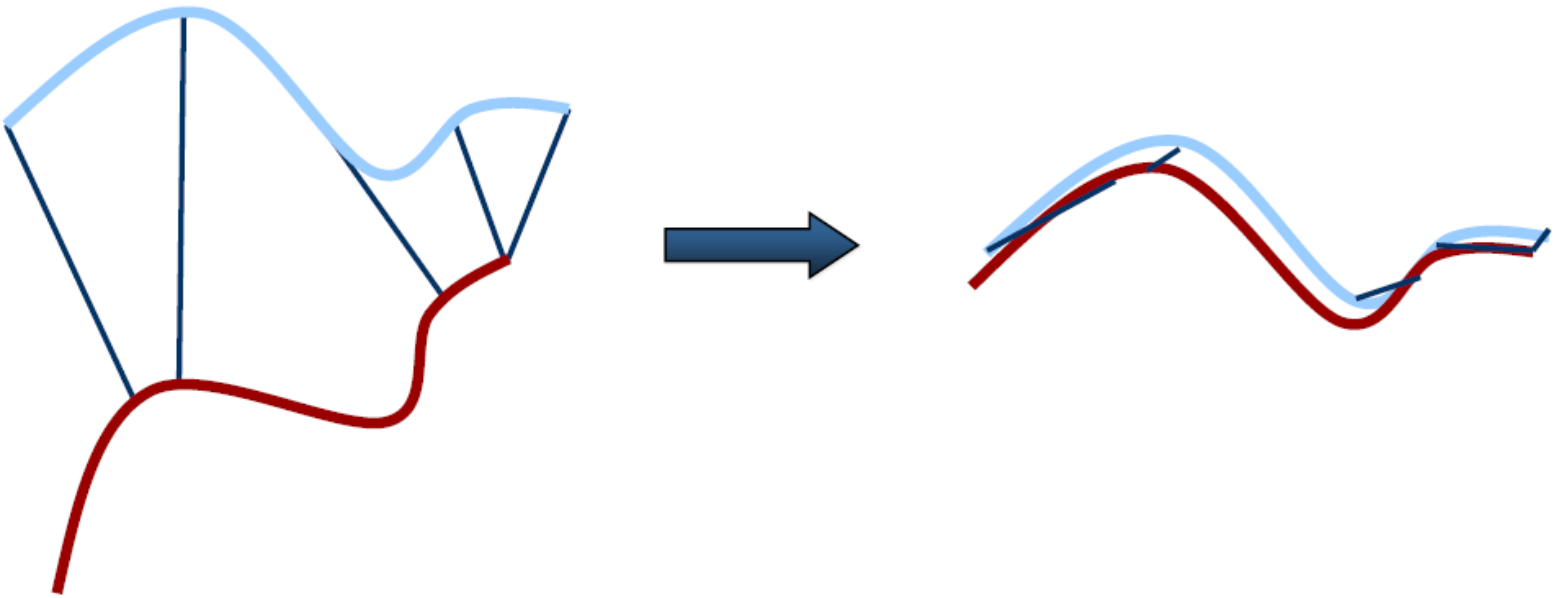
ICP: Key idea

- If the correct correspondences are known, the correct relative rotation/translation can be calculated in **closed form**



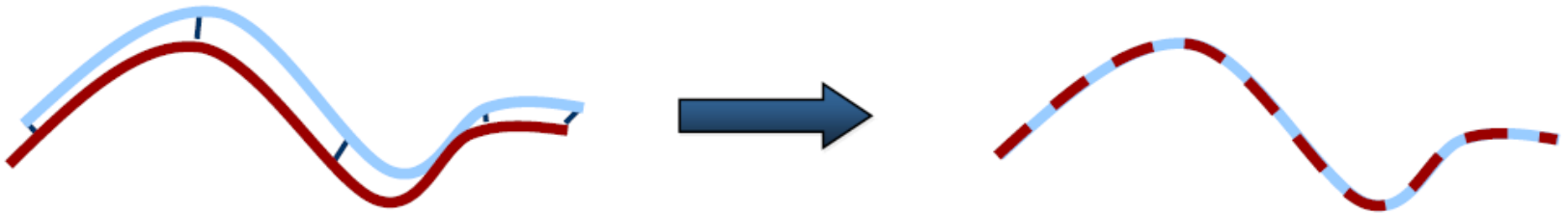
ICP: Iterative data association

- If the correct correspondences are **not known**, it is generally impossible to determine the optimal relative rotation and translation in one step



ICP: Iterative data association

- Idea: Iterate to find alignment
- Iterative Closest Points
  [Besl & McKay 92]
- Converges if starting positions are "close enough"



ICP: Iterative data association

- Determine corresponding points
- Compute rotation R, translation t via SVD
- Apply R and t to the points of the set to be registered
- Compute the error $E(R,t)$
- If error decreased and error > threshold
  - Repeat these steps
  - Stop and output final alignment, otherwise

ICP: Pseudocode

Input: Two point sets, X and P

Output: Translation t and rotation matrix R that best aligns the two point sets

ICP: Pseudocode

Input: Two point sets, *X* and *P*

Output: Translation *t* and rotation matrix *R*
that best aligns the two point sets

1. Start with a “good” alignment
2. Repeat until *t* and *R* are small:
3. **For every point in *X*, find its closest neighbor in *P***

ICP: Pseudocode

Input: Two point sets, *X* and *P*

Output: Translation *t* and rotation matrix *R*
that best aligns the two point sets

1. Start with a “good” alignment
2. Repeat until *t* and *R* are small:
3. **For every point in *X*, find its closest neighbor in *P***
4. Find min *t* and *R* for that correspondence assignment
5. Translate and rotate *P* by *t* and *R*

ICP: Pseudocode

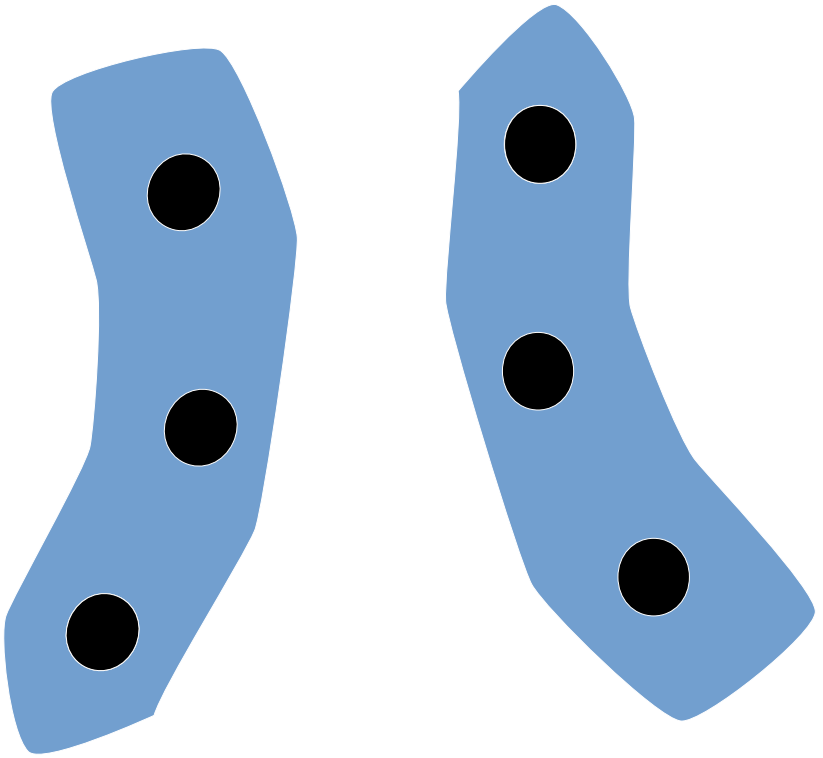
Input: Two point sets, *X* and *P*

Output: Translation *t* and rotation matrix *R*
that best aligns the two point sets

1. Start with a “good” alignment
2. Repeat until *t* and *R* are small:
3. **For every point in *X*, find its closest neighbor in *P***
4. Find min *t* and *R* for that correspondence assignment
5. Translate and rotate *P* by *t* and *R*
6. Return the net translation *t* and rotation matrix *R*

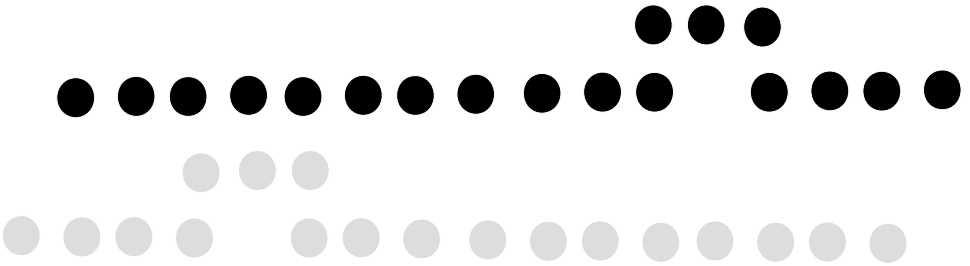
Converges if the point sets are initially well-aligned
(Besl & McKay, 1992)

Question



Where does ICP converge for this initial configuration?

Question



How does ICP align these two point sets?

Iterative closest point (ICP)

Variants on the following stages of ICP have been proposed:

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Iterative closest point (ICP)

- ICP is a powerful algorithm for calculating the displacement between scans
- The major problem is to determine the correct data associations
- Convergence speed depends on point matched points
- Given the correct data associations, the transformation can be computed efficiently using SVD
- ICP does not always converge

Outline

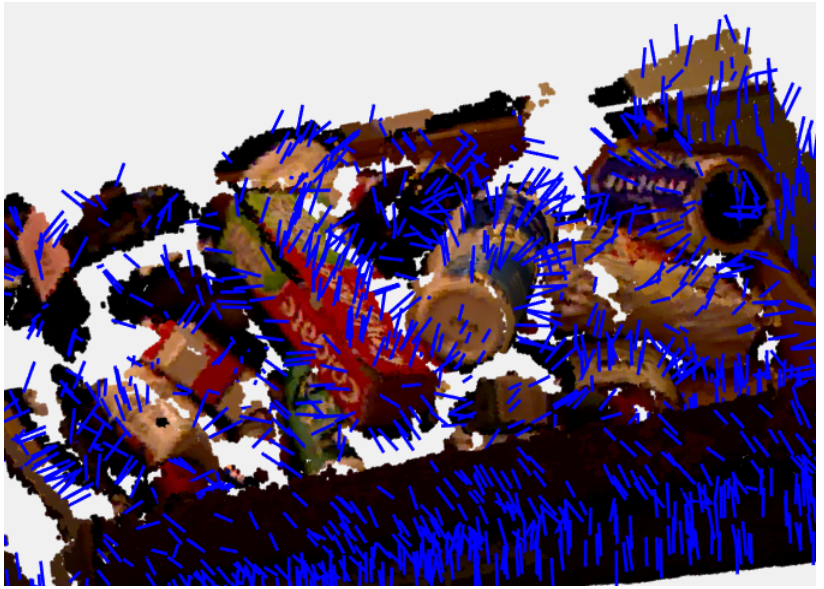
✔ Aligning point clouds
- Iterative closest point

Extracting primitive shapes
- Surface normals
- Planes
- RANSAC
- More shapes (time permitting)

Advanced perception (time permitting)

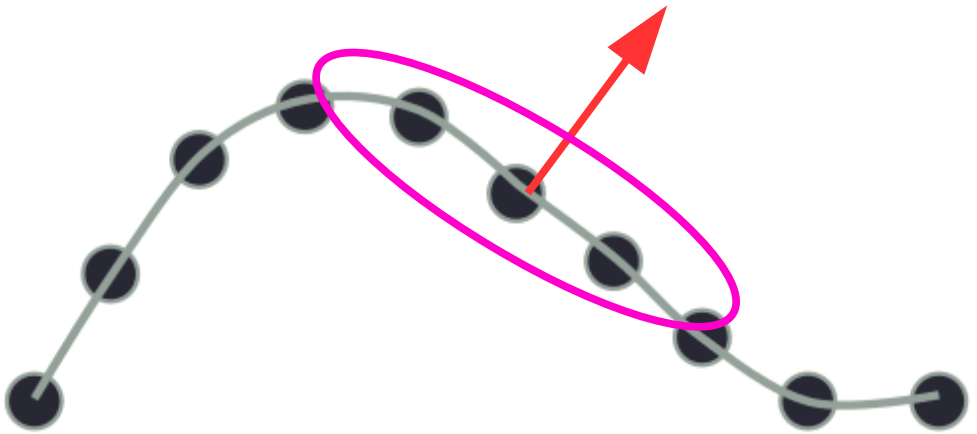
Surface normals

Point cloud



Point cloud w/ surface normals
(normals are subsampled)

Surface normals



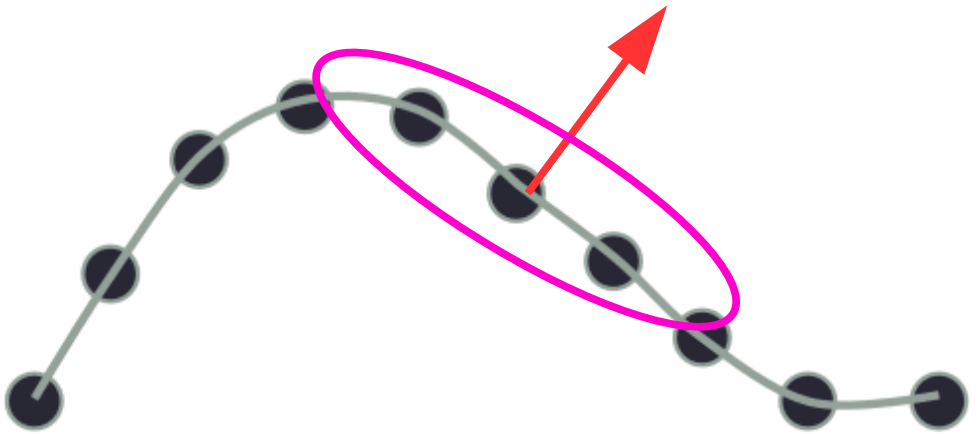
How do we calculate the surface normal given only points?

Answer:

1. Calculate the sample covariance matrix of the points within a local neighborhood of the surface normal

2. Take eigenvalues/eigenvectors of that covariance matrix

Surface normals



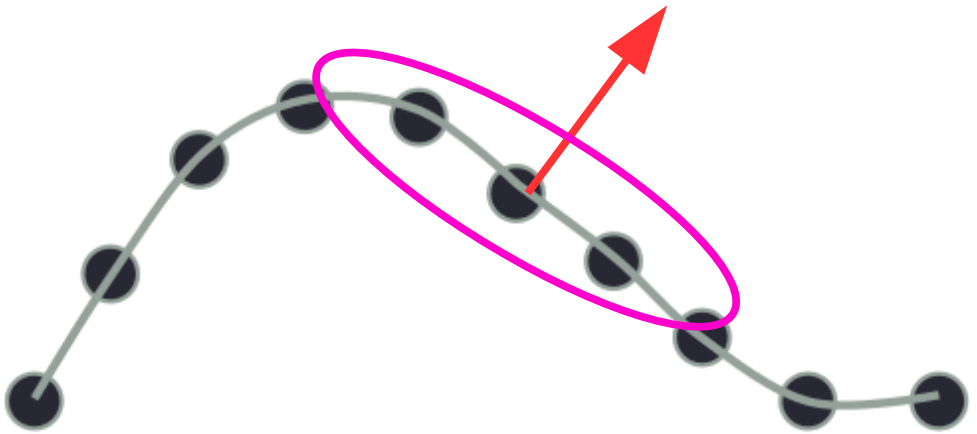
Let *C* denote the set of points in the point cloud

Suppose we want to calculate the surface normal at $x \in C$

Let $B_r(x) \subseteq \mathbb{R}^3$ denote the r-ball about *x*

$N_r(x) = B_r(x) \cap C$ is the set of points in the cloud within *r* of *x*

Surface normals

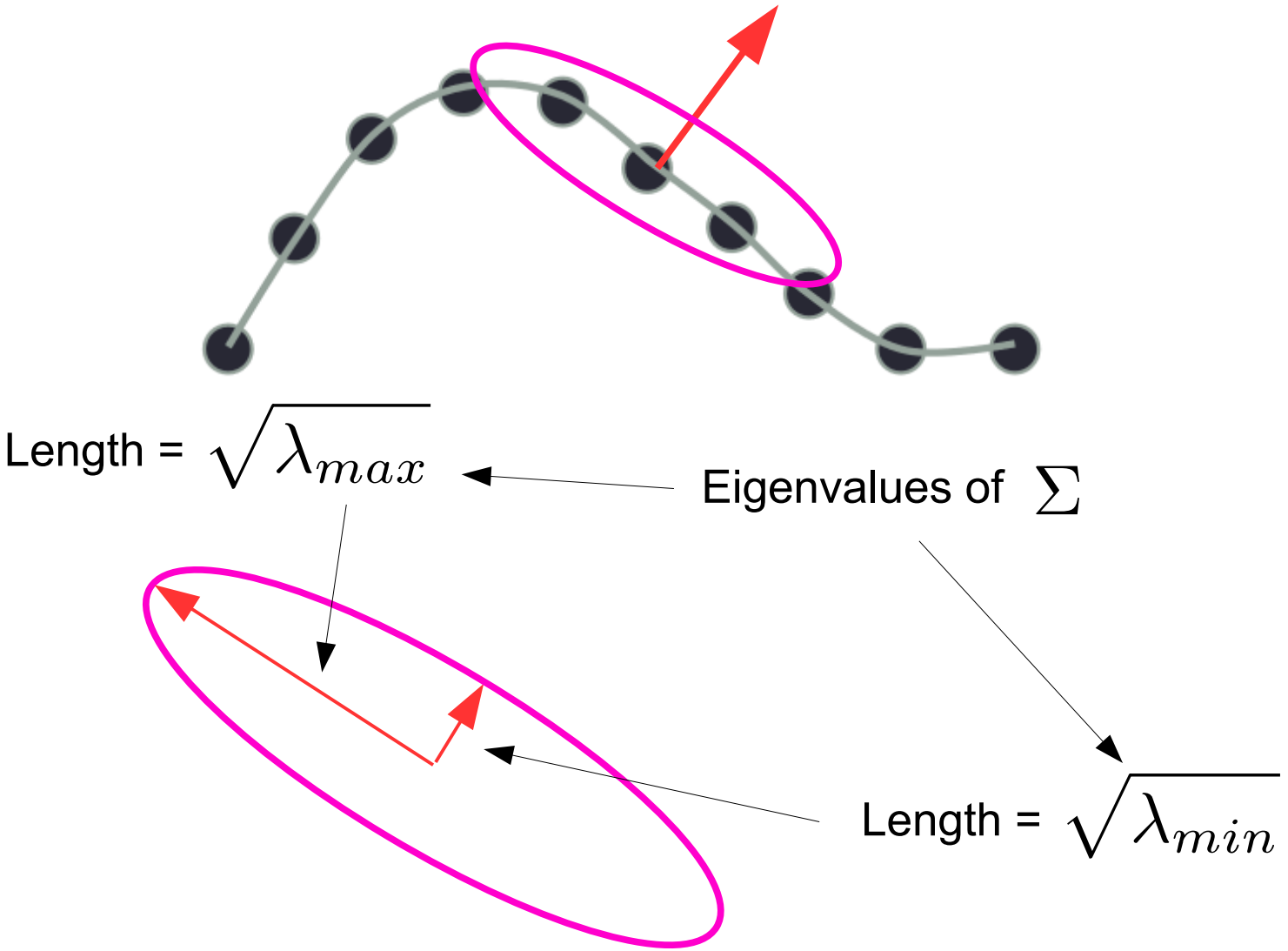


Calculate the sample covariance matrix of the points in $N_r(x)$

$$\Sigma = \sum_{p \in N_r(x)} (p - \bar{p})(p - \bar{p})^T$$

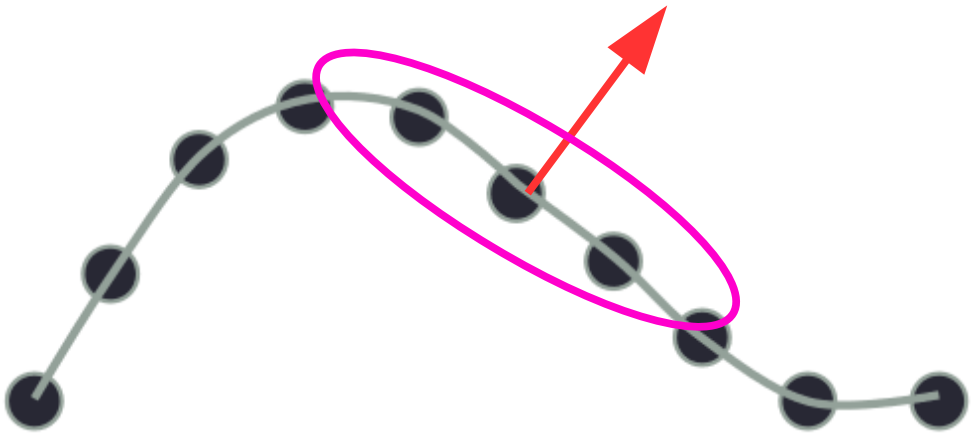
$$\bar{p} = \frac{1}{|N_r(x)|} \sum_{p \in N_r(x)} p$$

Surface normals



Principle axes of ellipse point in directions of corresponding eigenvectors

Surface normals



Surface normal is in the direction of the eigenvector corresponding to the smallest eigenvalue of $\sum$

There should be two large eigenvalues and one small eigenvalue.

Summary: Computing surface normals

1. Calculate points within r-ball about x: $N_r(x) = B_r(x) \cap C$

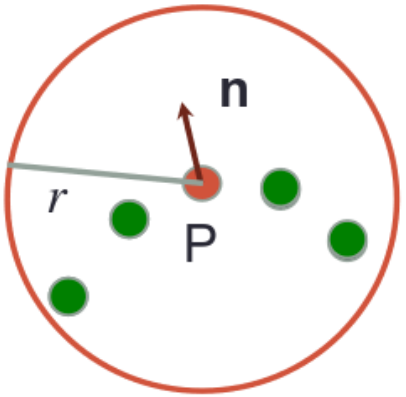
2. Calculate covariance matrix: $$\Sigma = \sum_{p \in N_r(x)} (p - \bar{p})(p - \bar{p})^T$$

3. Calculate eigenvectors: v_1, v_2, v_3
   and eigenvalues: $\lambda_1, \lambda_2, \lambda_3$ (λ_3 is smallest)

4. v_3 is parallel or anti-parallel to surface normal

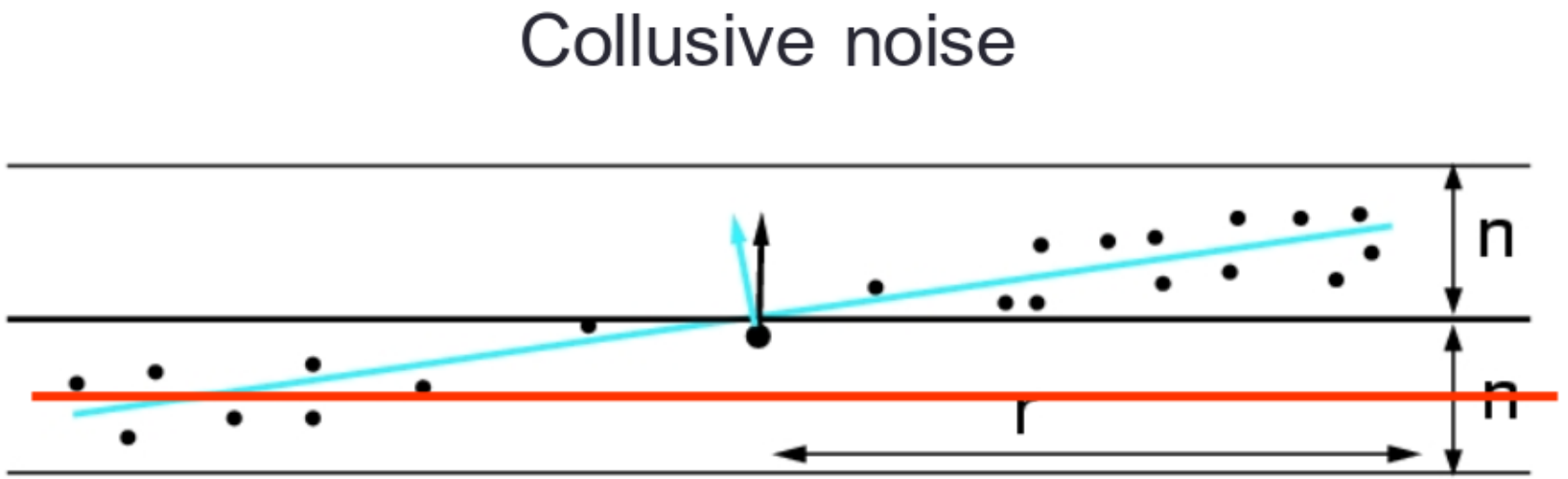
Calculating surface normals

How large a point neighborhood to use when calculating $\sum$?



Because points can be uneven, do not use k-nearest neighbor.
– Select a radius *r*
– Which what value of *r* to use?

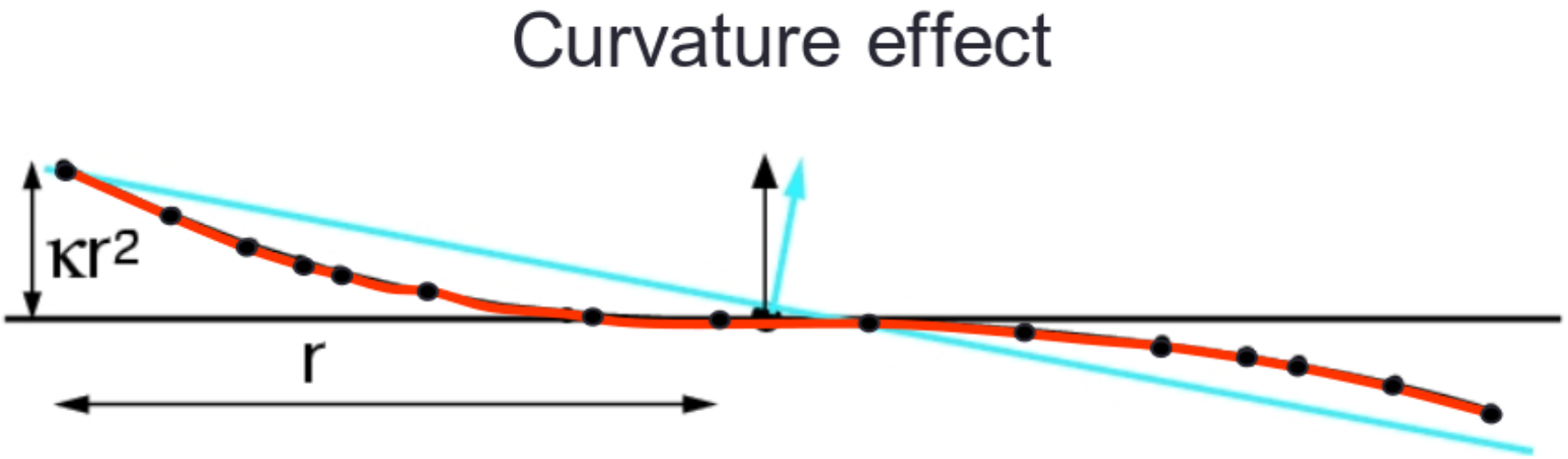
Calculating surface normals



Because of noise in the data, small *r* may lead to underfitting.

Images from Course INF 555 slides, Ecole Polytechnique, Paris

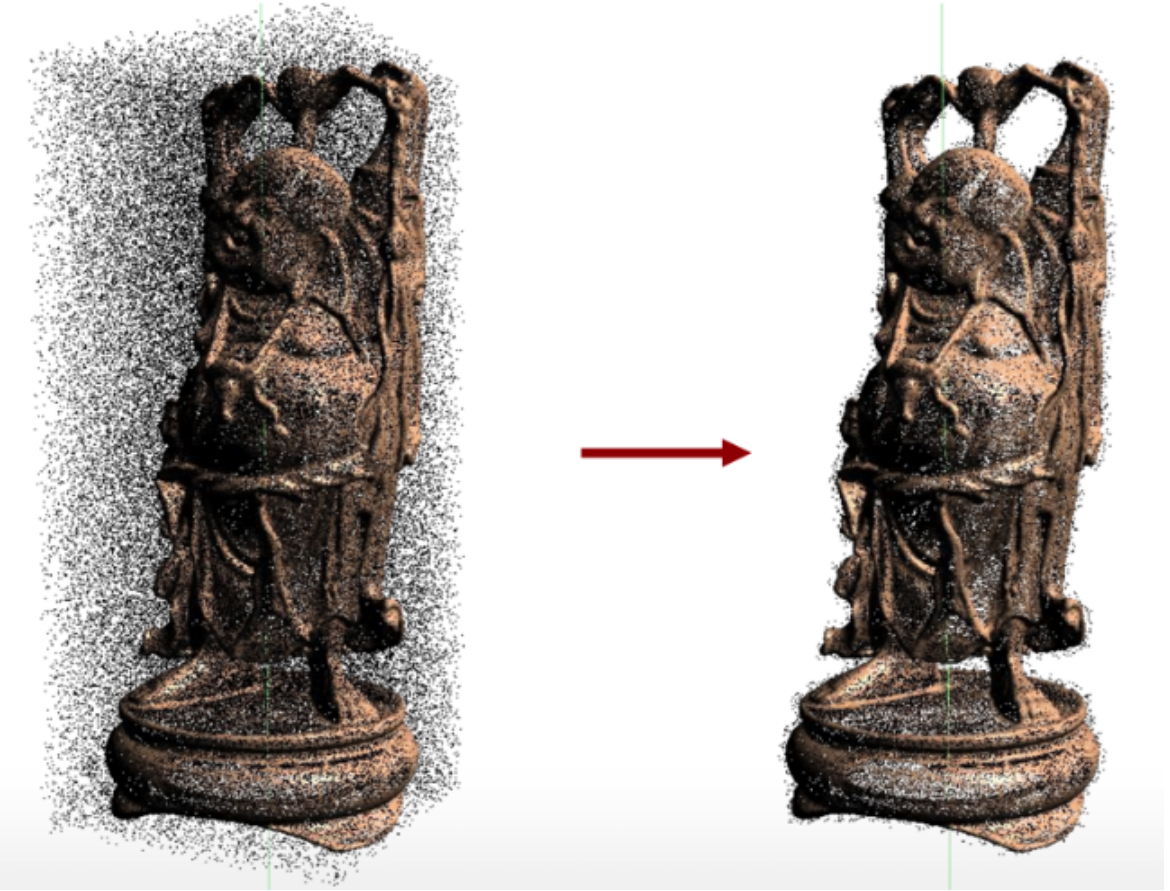
Calculating surface normals



Due to curvature, large *r* can lead to estimation bias.

Images from Course INF 555 slides, Ecole Polytechnique, Paris

Application: Outlier removal



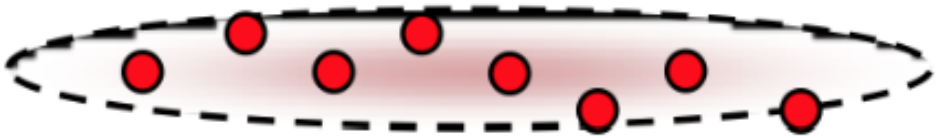
Similar approach as in normal estimation:

1. Calculate local covariance matrix
2. Estimate eigenvectors / eigenvalues
3. Use that information somehow ...

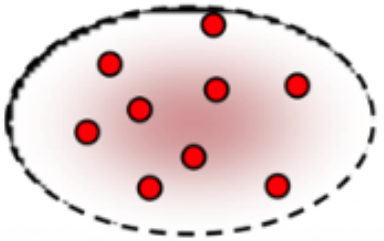
Images from Course INF 555 slides, Ecole Polytechnique, Paris

Application: Outlier removal

If points lie on a plane or line, then $\frac{\lambda_{min}(\Sigma)}{\lambda_{max}(\Sigma)}$ is small



If points are uniformly random, then $\frac{\lambda_{min}(\Sigma)}{\lambda_{max}(\Sigma)}$ is close to 1

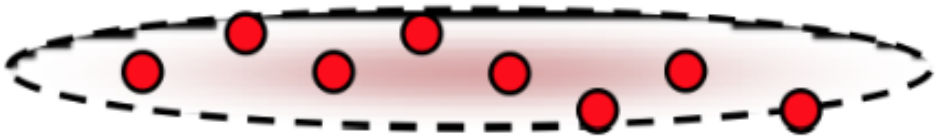


Outlier removal: delete all points for which $\frac{\lambda_{min}(\Sigma)}{\lambda_{max}(\Sigma)}$ is above a threshold

Images from Course INF 555 slides, Ecole Polytechnique, Paris

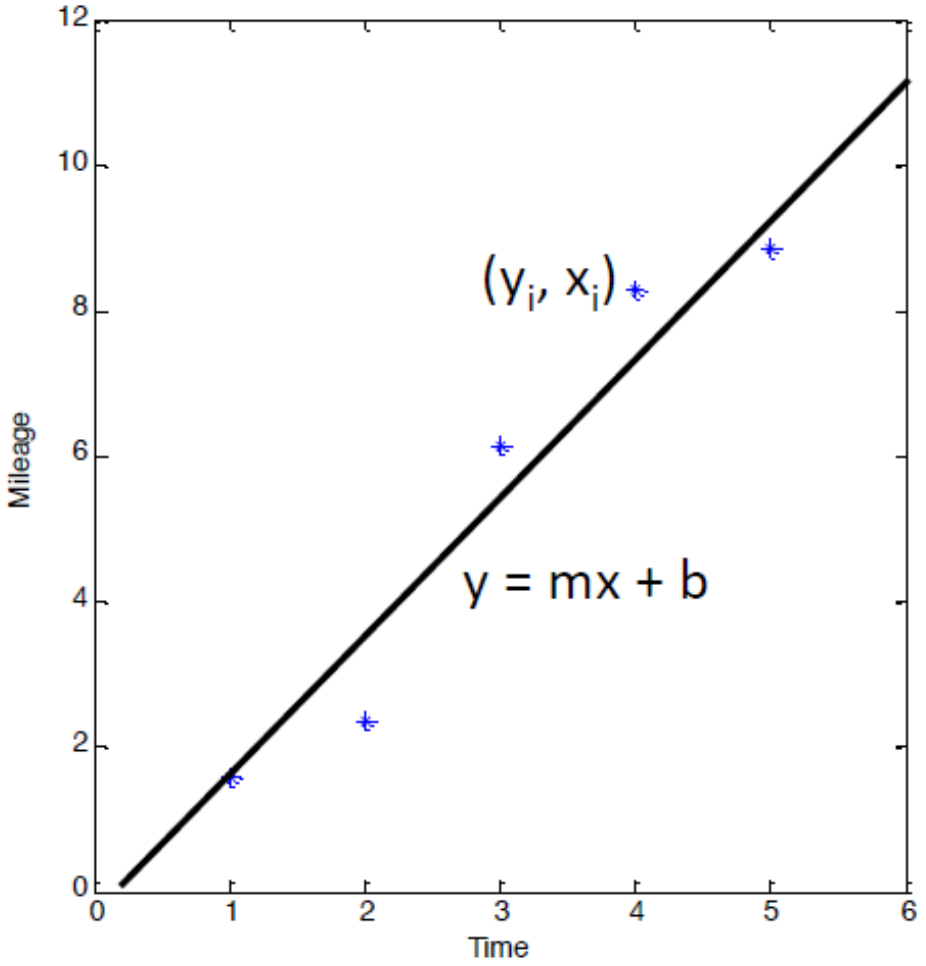
Application: Plane detection

If points lie on a plane or line, then $\frac{\lambda_{min}(\Sigma)}{\lambda_{max}(\Sigma)}$ is small

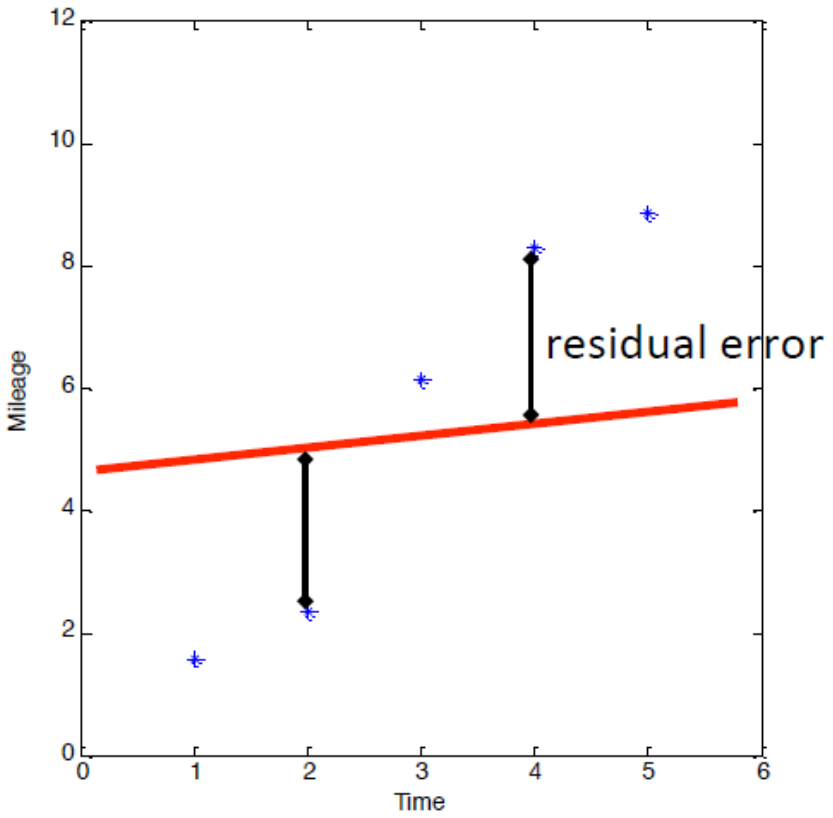


How to determine which plane(s) is the best fit?

RANSAC

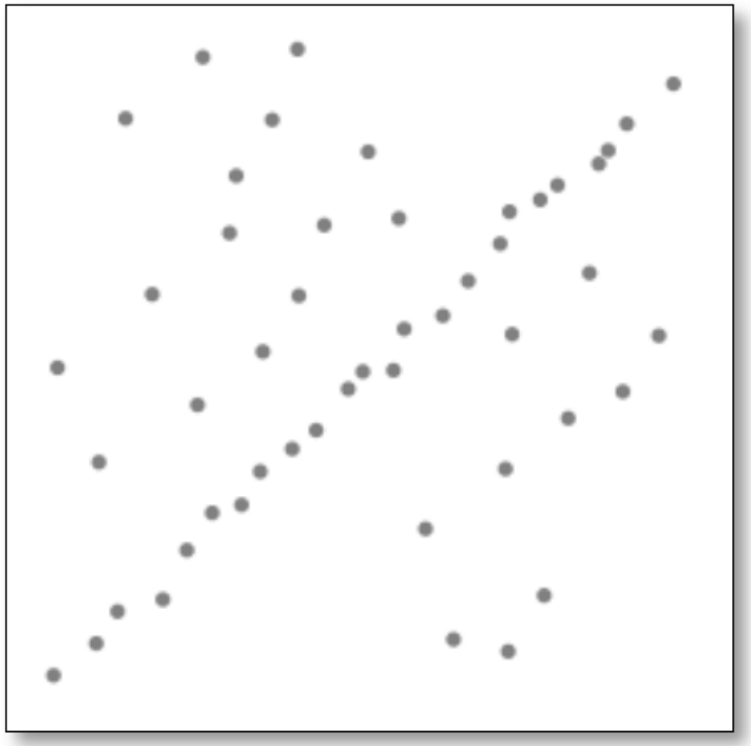


RANSAC

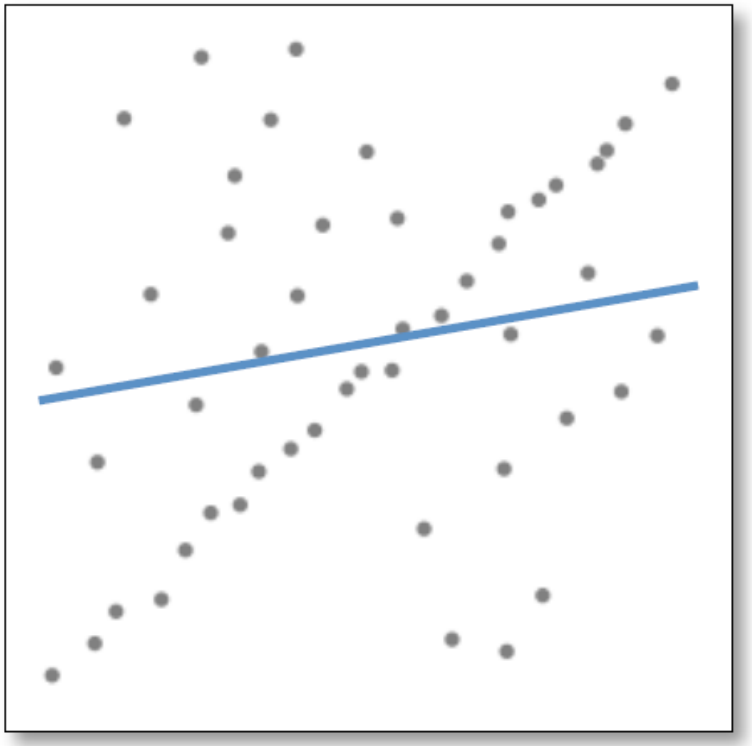


$$\text{Cost}(m, b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2$$

RANSAC



Problem: Fit a line to these datapoints



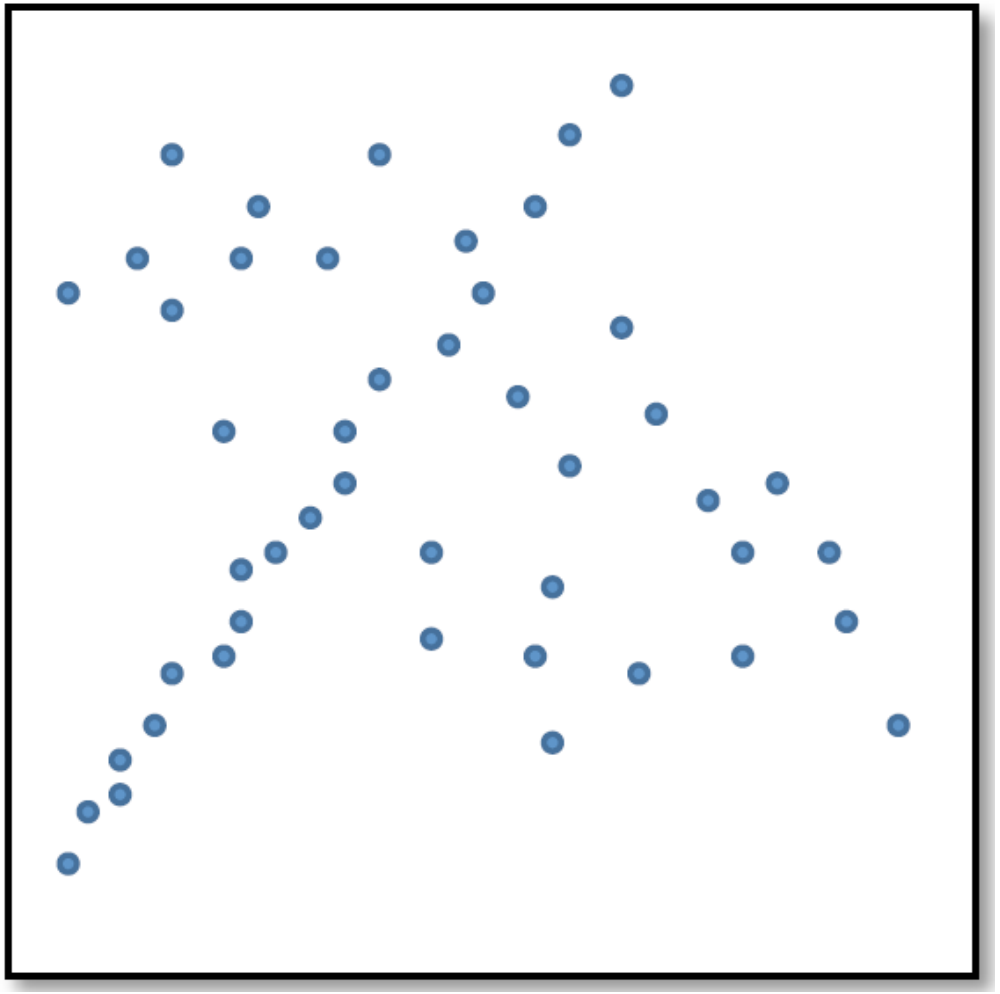
Least squares fit

RANSAC: Key idea

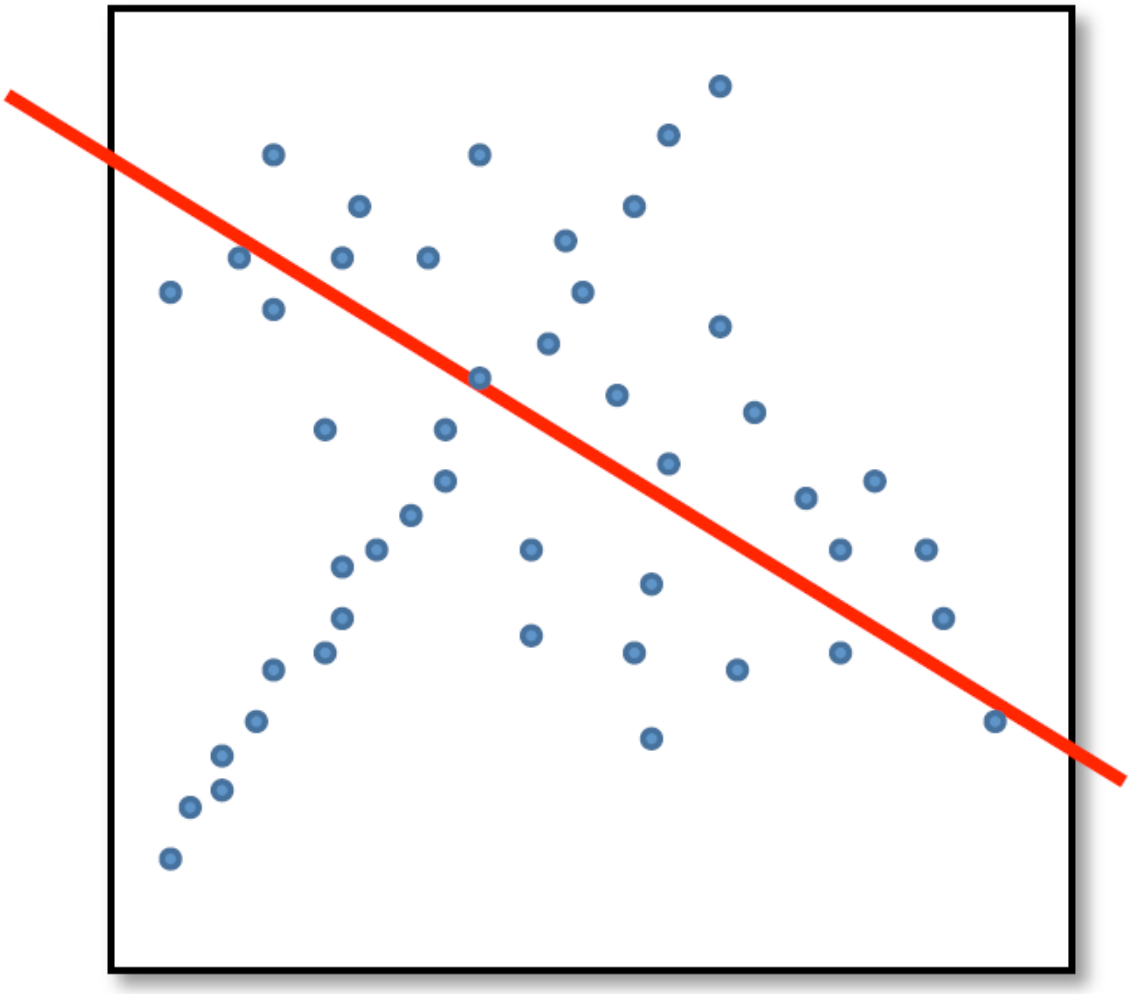
- Given a hypothesized line
- Count the number of points that "agree" with the line
  - "Agree" = within a small distance of the line
  - I.e., the **inliers** to that line

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

Counting inliers

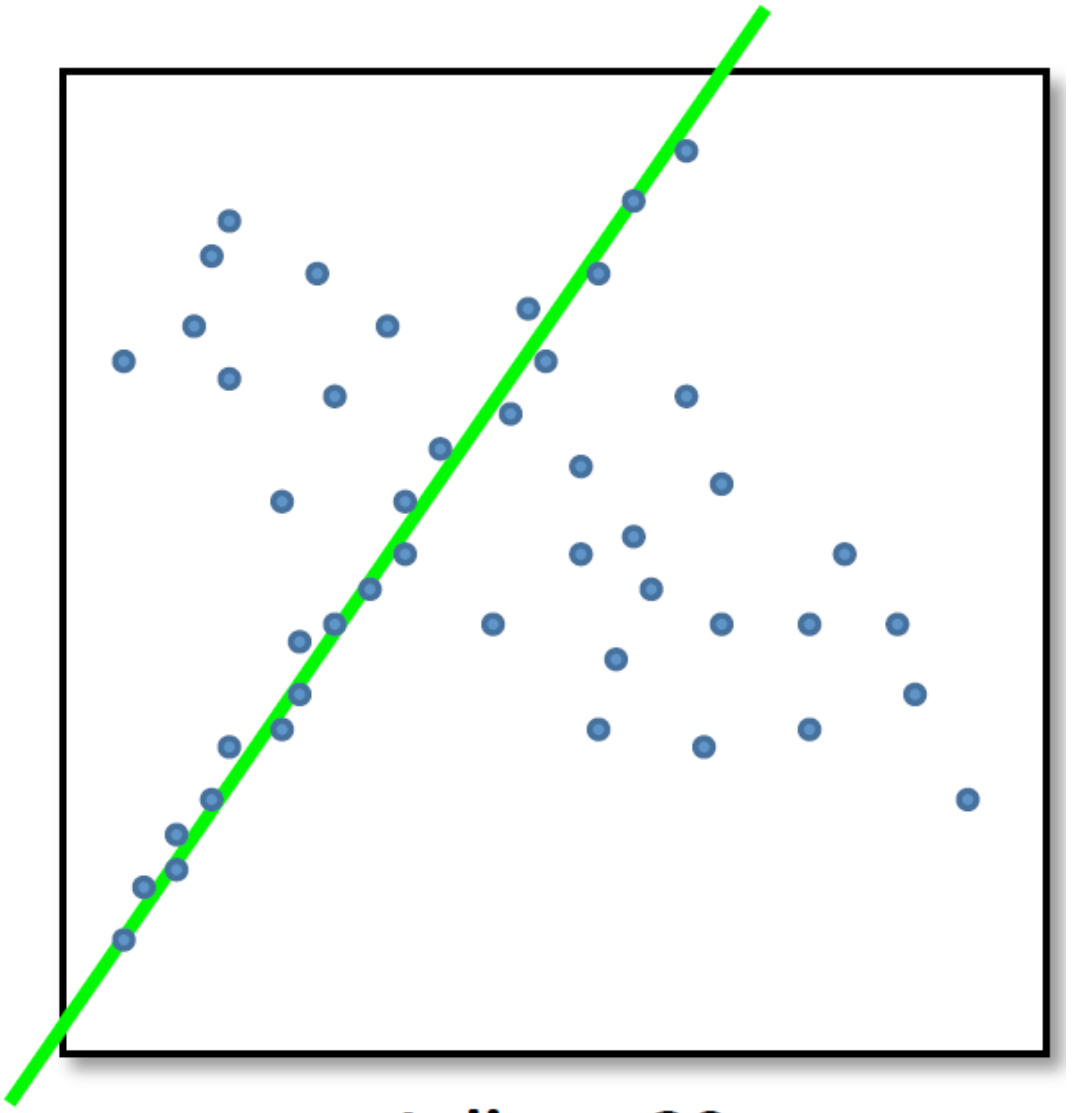


Counting inliers



Inliers: 3

Counting inliers



Inliers: 20

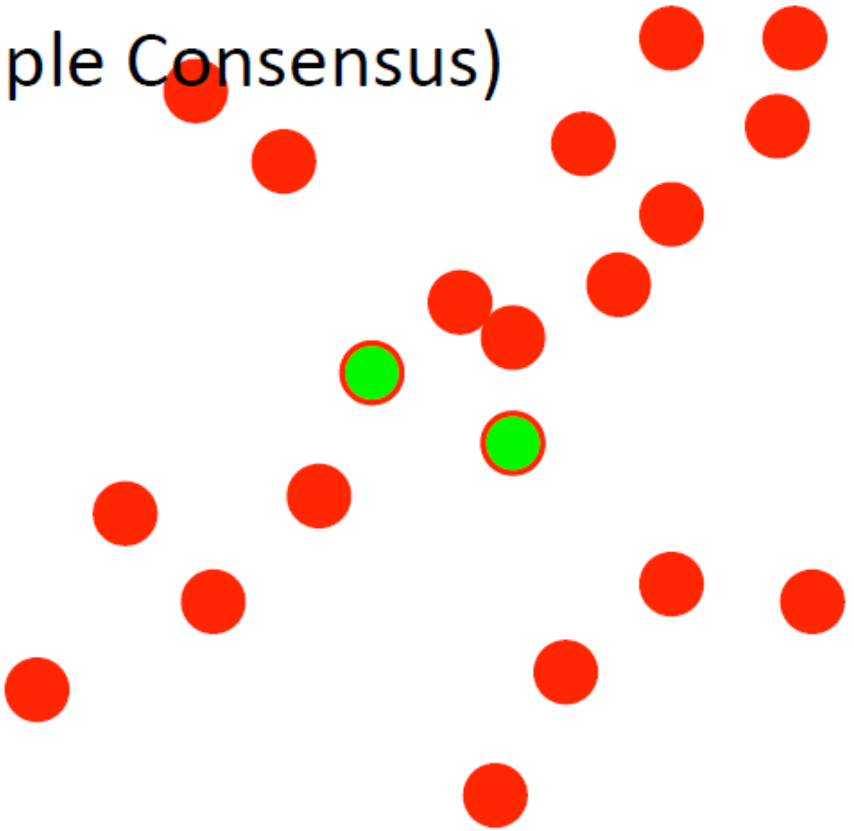
Finding the best fit

- Unlike least-squares, no simple closed-form solution

- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

RANSAC (Random Sample Consensus)

Line fitting example



Algorithm:

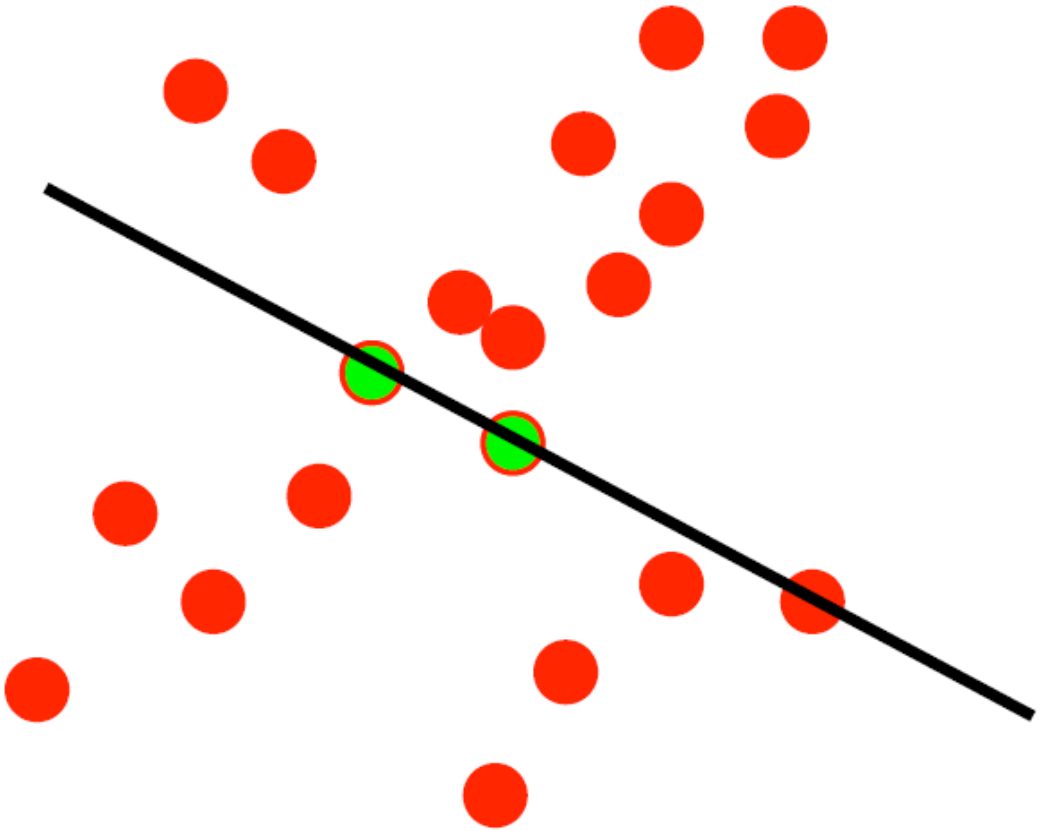
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example



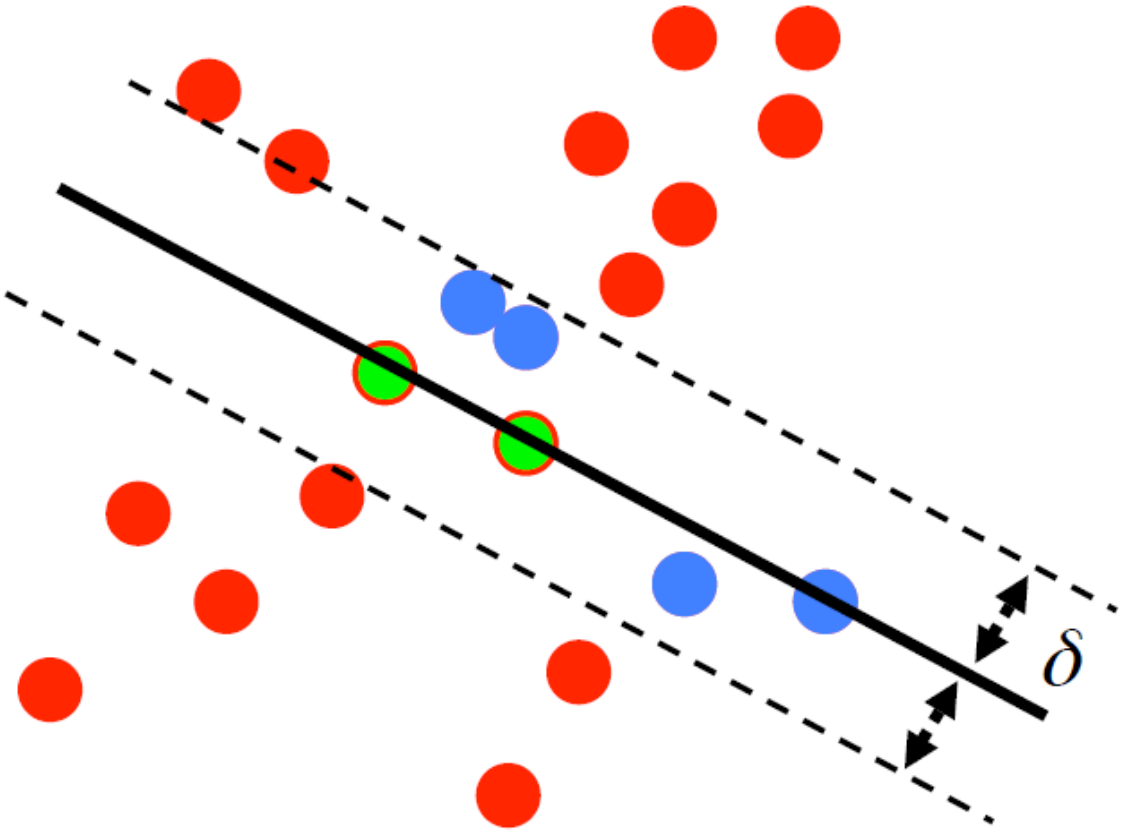
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
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

RANSAC

Line fitting example



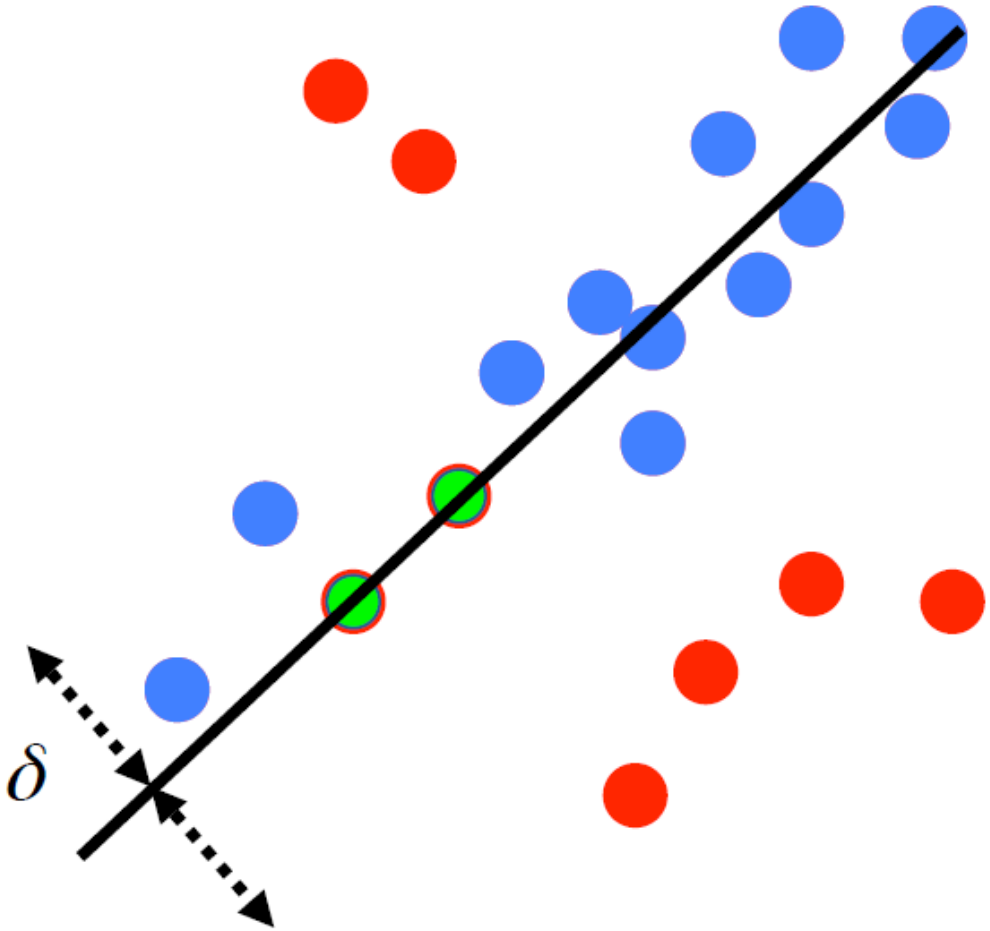
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
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

RANSAC



$N_I = 14$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
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

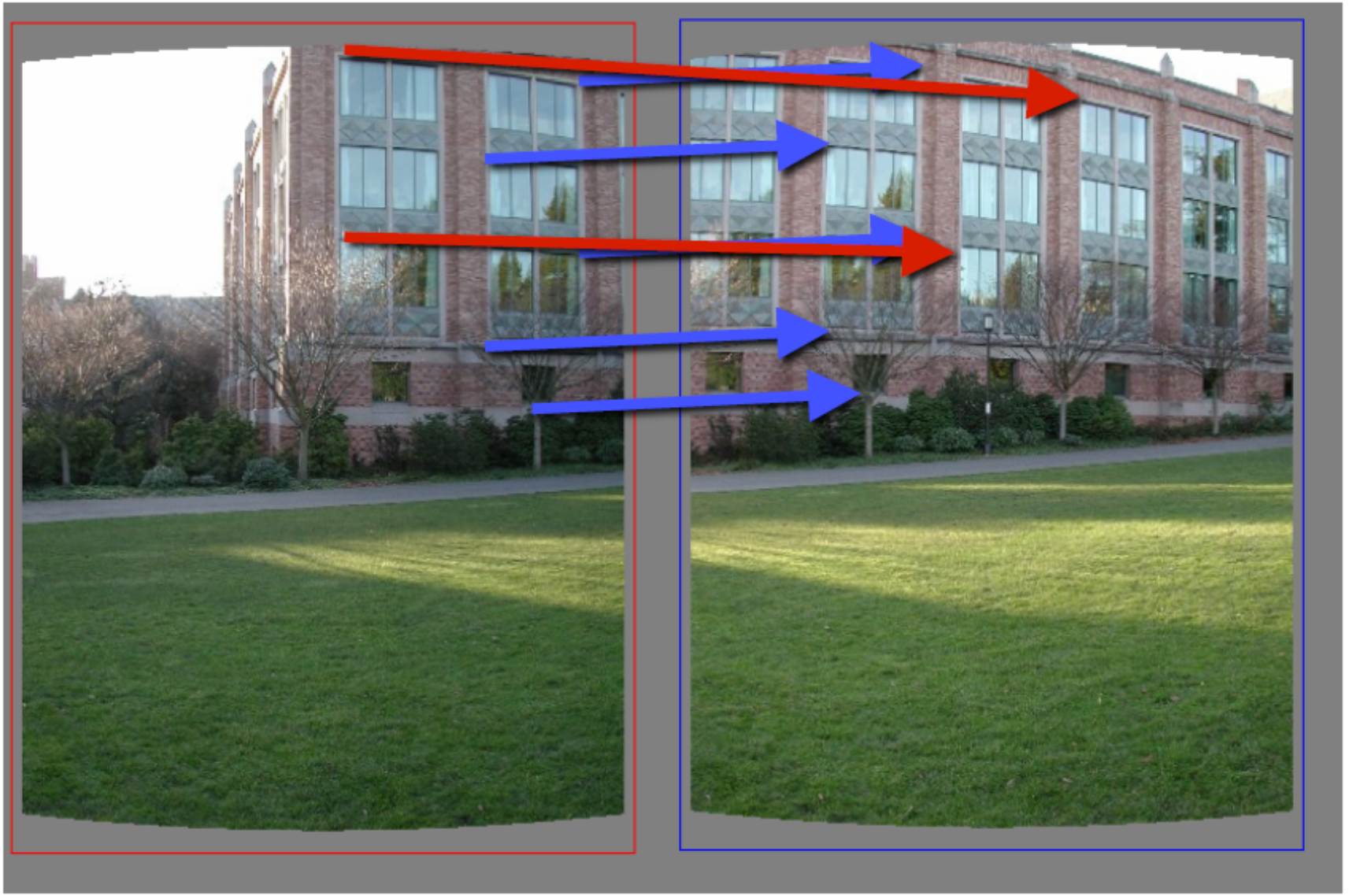
RANSAC: Key idea

- 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

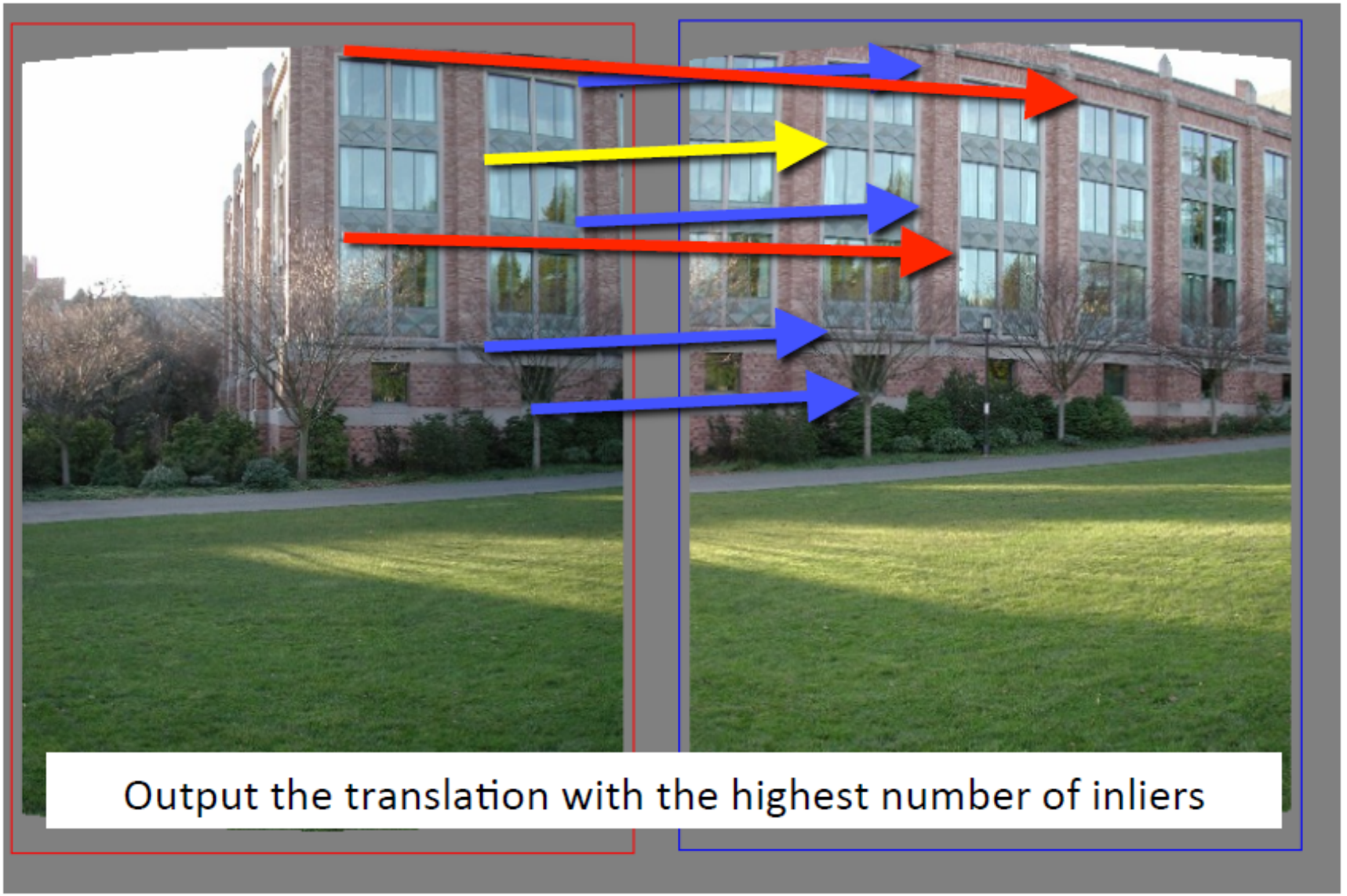
- "All good matches are alike; every bad match is bad in its own way."

  – Tolstoy via Alyosha Efros

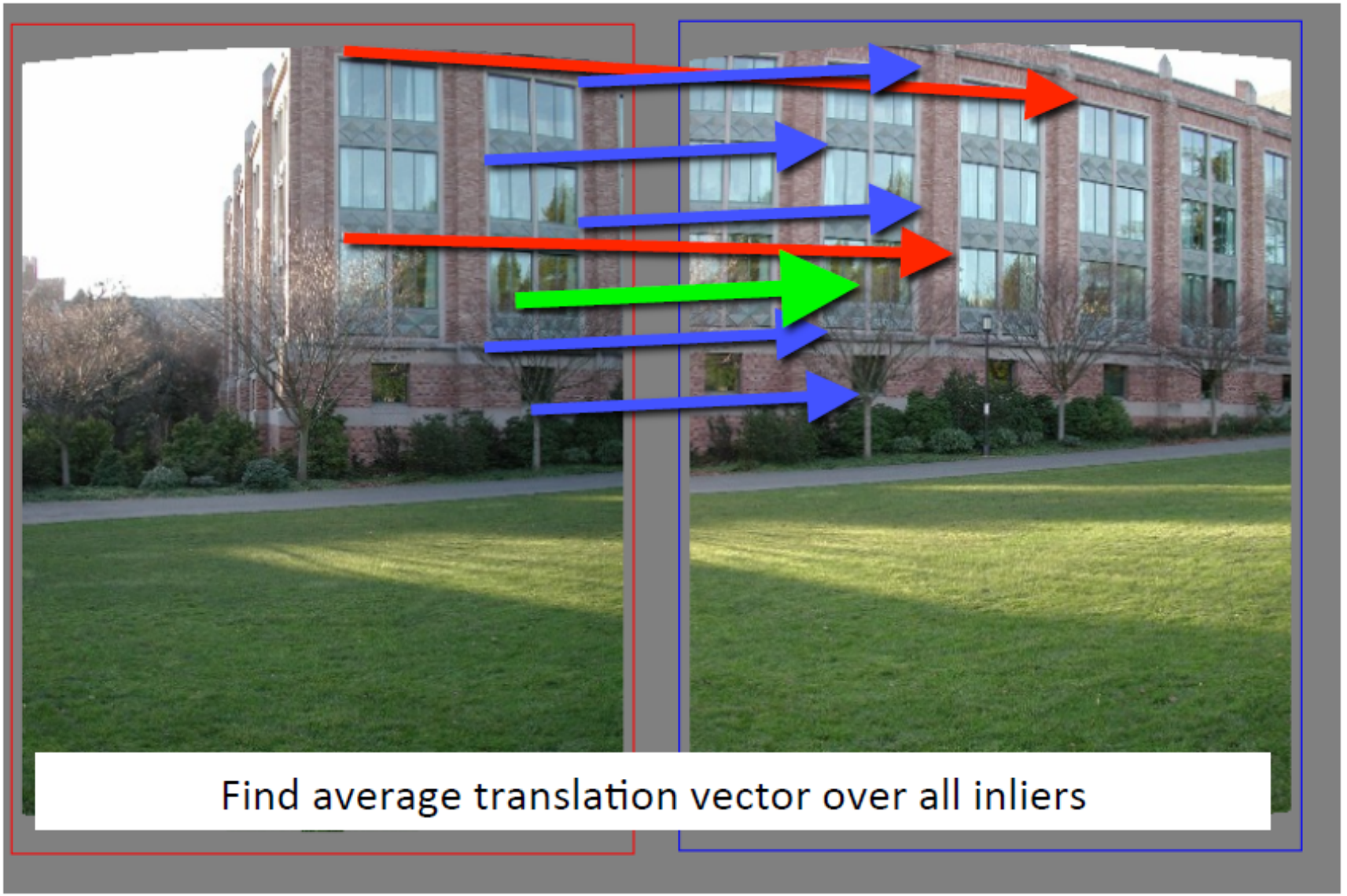
Application: Image matching



Application: Image matching



Application: Image matching



RANSAC: Hyperparameters

- **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: Hyperparameters

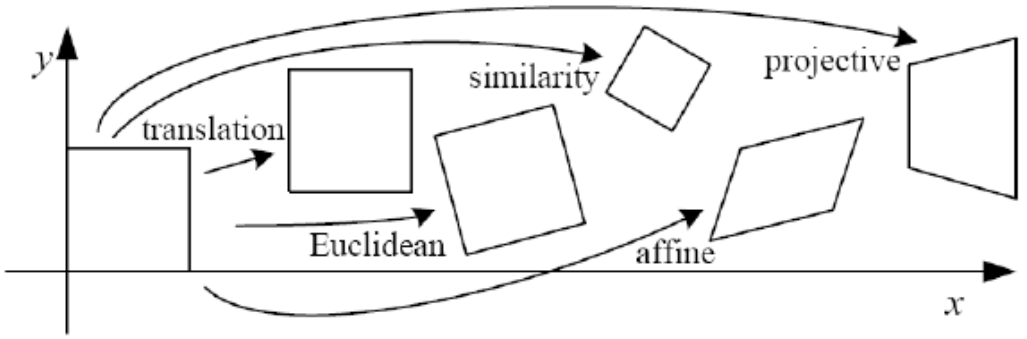
- If we have to choose *k* samples each time
  - with an inlier ratio p
  - and we want the right answer with probability *P*

	proportion of inliers *p*						
k	95%	90%	80%	75%	70%	60%	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

$P = 0.99$

RANSAC: What is *k* ?

- 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	$\left[\begin{array}{c\|c} \boldsymbol{I} & \boldsymbol{t} \end{array}\right]_{2\times3}$	2	orientation + ···	
rigid (Euclidean)	$\left[\begin{array}{c\|c} \boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	3	lengths + ···	
similarity	$\left[\begin{array}{c\|c} s\boldsymbol{R} & \boldsymbol{t} \end{array}\right]_{2\times3}$	4	angles + ···	
affine	$\left[\begin{array}{c} \boldsymbol{A} \end{array}\right]_{2\times3}$	6	parallelism + ···	
projective	$\left[\begin{array}{c} \tilde{\boldsymbol{H}} \end{array}\right]_{3\times3}$	8	straight lines	

RANSAC: Summary

- 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

RANSAC: Summary

- Least Squares Fit
  - closed form solution
  - robust to noise
  - not robust to outliers
- Hough transform
  - robust to noise and outliers
  - can fit multiple models
  - only works for a few parameters (1-4 typically)
- RANSAC
  - robust to noise and outliers
  - works with a moderate number of parameters (e.g, 1-8)

Outline

- ✔ Aligning point clouds
  - Iterative closest point

- ✔ Extracting primitive shapes
  - Surface normals
  - Planes
  - RANSAC
  - More shapes (time permitting)

Advanced perception (time permitting)

Feedback

Piazza thread: 4/6 Lec 19 Feedback

Please post your answers to the following anonymously.

1. What did you like today?

2. What was unclear?

3. Thoughts on Monday's (second) ethics lecture?

4. Any progress on the robots?

5. Any additional feedback / comments?