Computer Vision

Lec 4 - RANSAC

University of Haifa

Simon Korman

Robust Fitting using RANSAC

slide credit

- Svetlana Lazebnik, Sanja Fidler, Kristen Grauman, Ioannis Gkioulekas, Kris Kitani, James Hays, Fredo Durand, Rick Szeliski, Andrew Zisserman, Kyros Kutulakos

Overview

- Interest point detection
  - Harris, LoG, Scale space
- Invariant descriptors
  - SIFT
- Basic (nearest neighbor based) matching
  - ratio test, bidirectional criterions
- Fitting of 2D transformations (models)
  - affine, homography
- Today: RANSAC for *robust* fitting

Motivation: mosaics

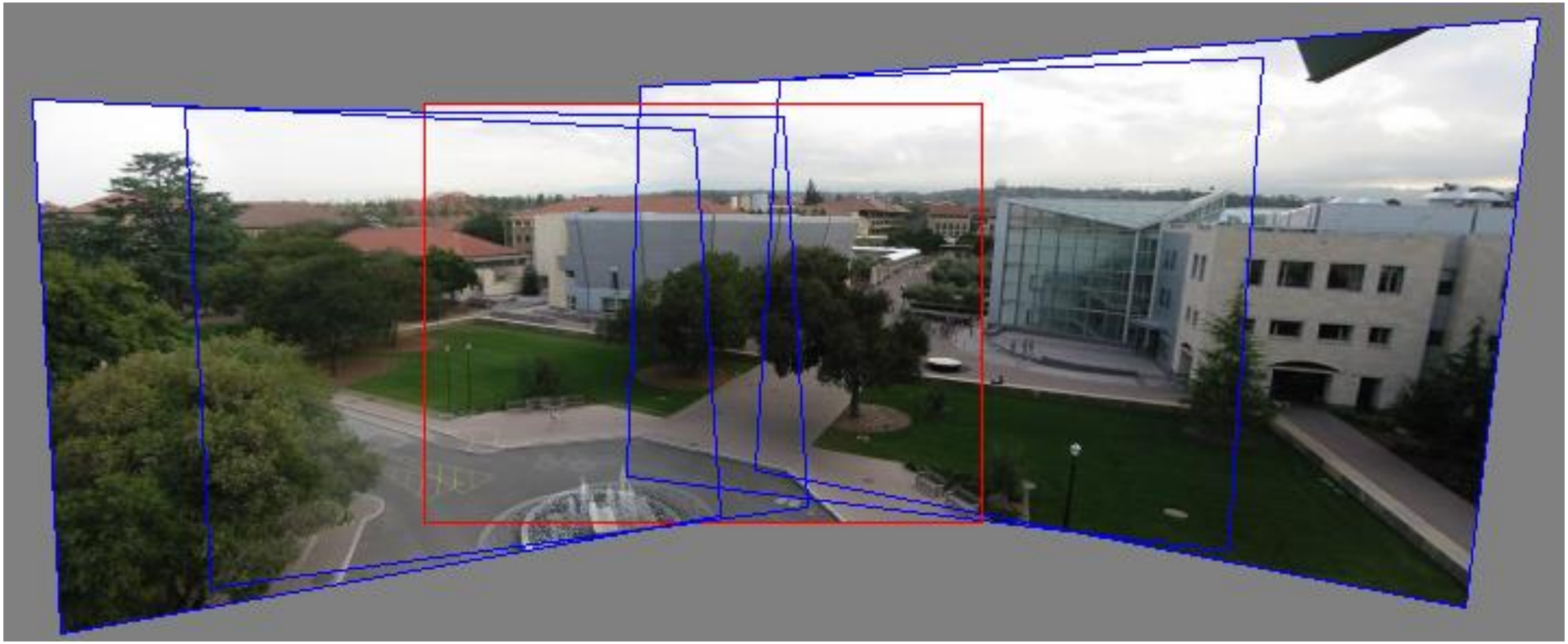
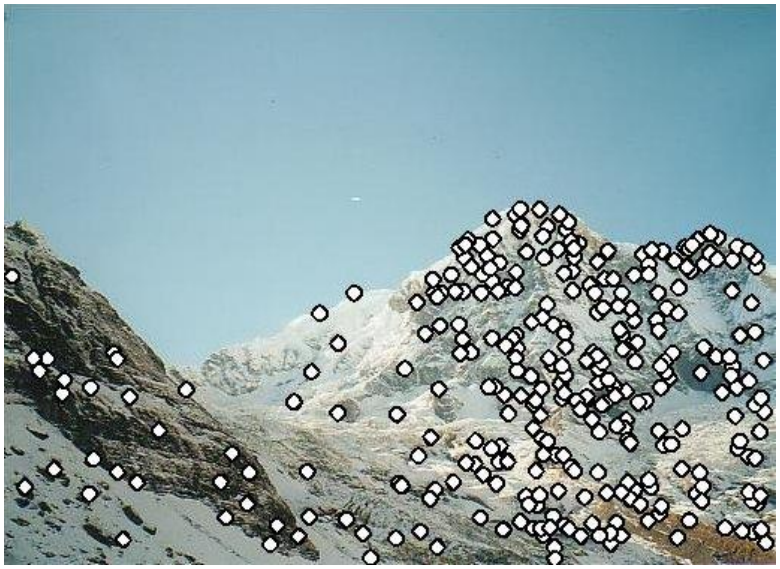


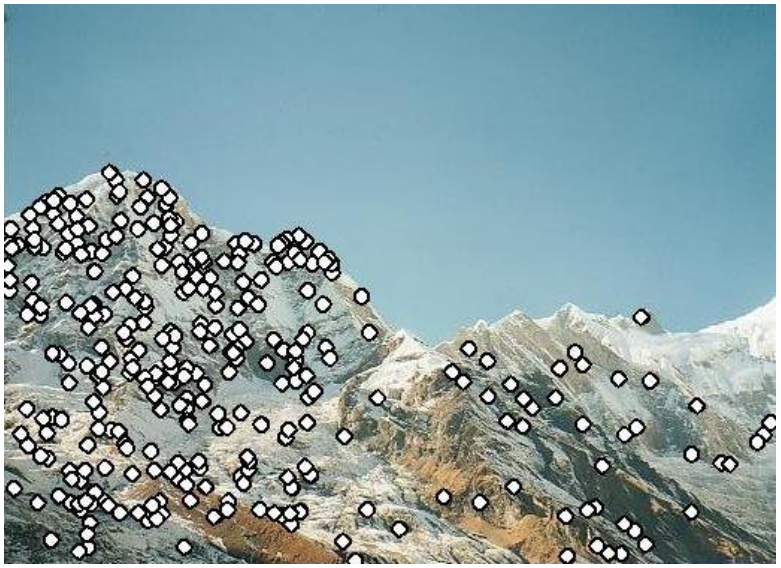
Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fal

Robust feature-based alignment

Source: L. Lazebnik

Robust feature-based alignment





- Extract features

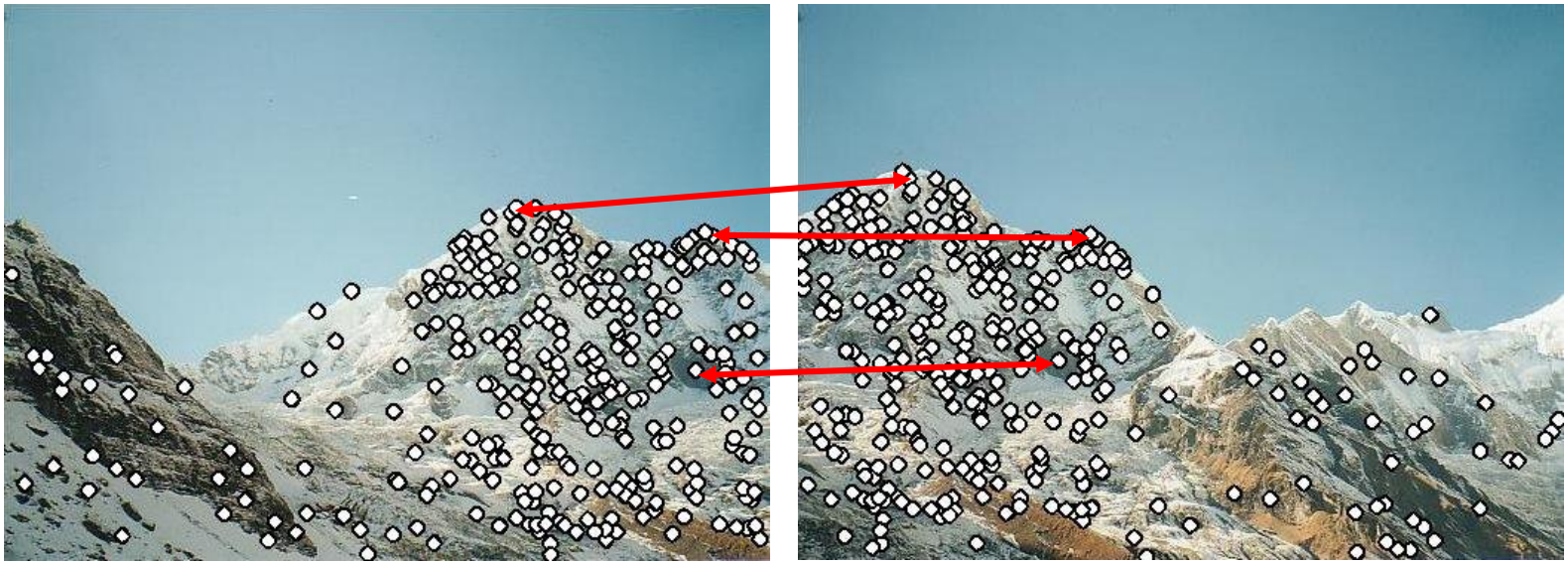
Source: L. Lazebnik

Robust feature-based alignment

- Extract features
- Compute *putative matches*

Source: L. Lazebnik

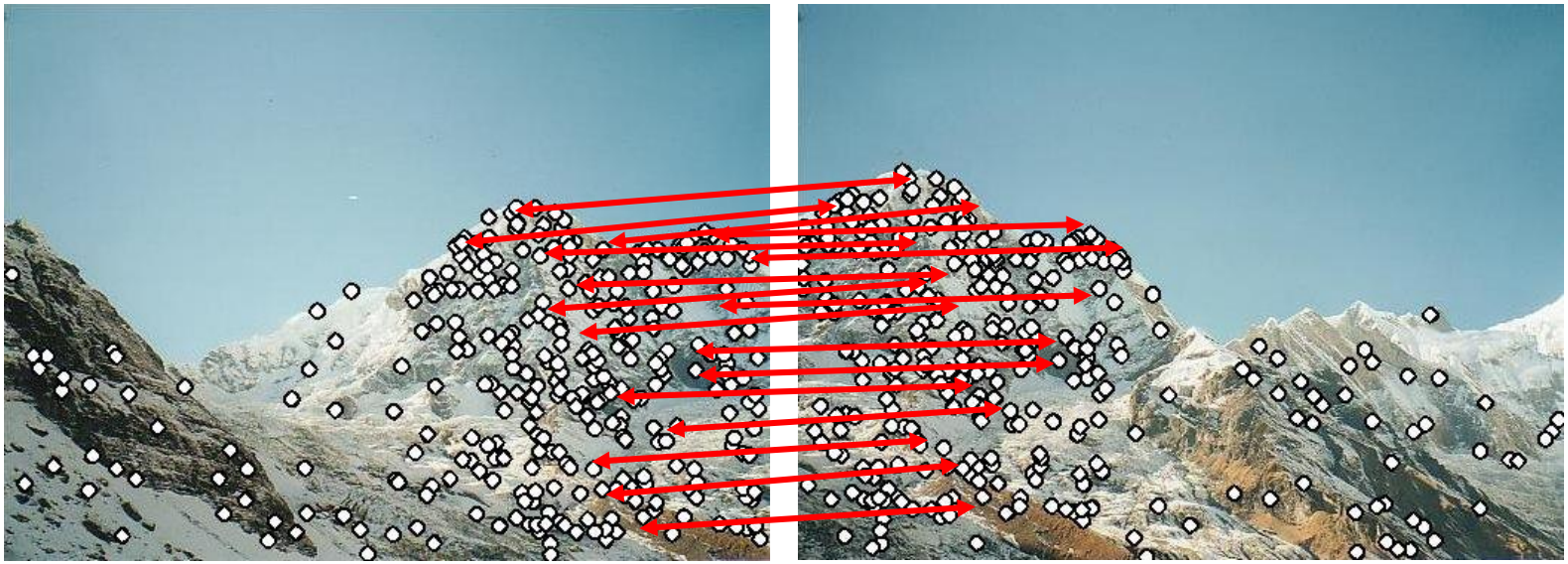
Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

Source: L. Lazebnik

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

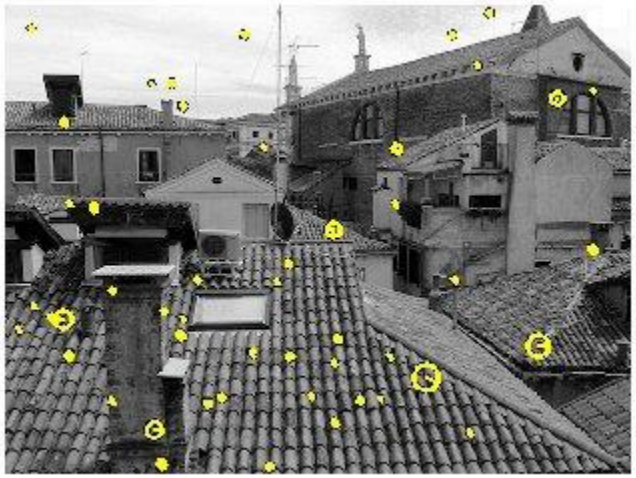
Source: L. Lazebnik

Robust feature-based alignment

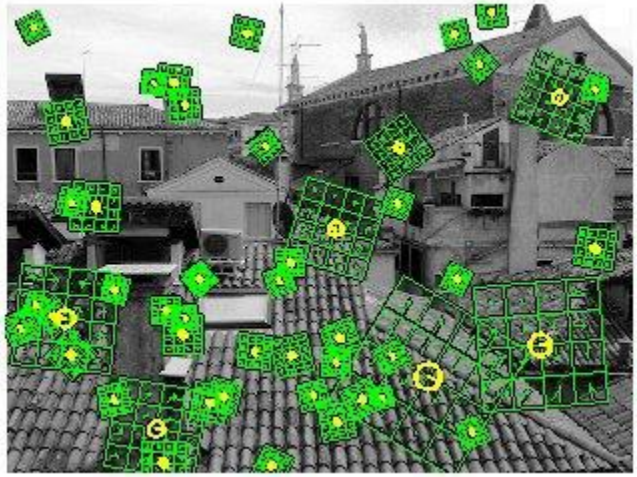
- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

Recall: Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]



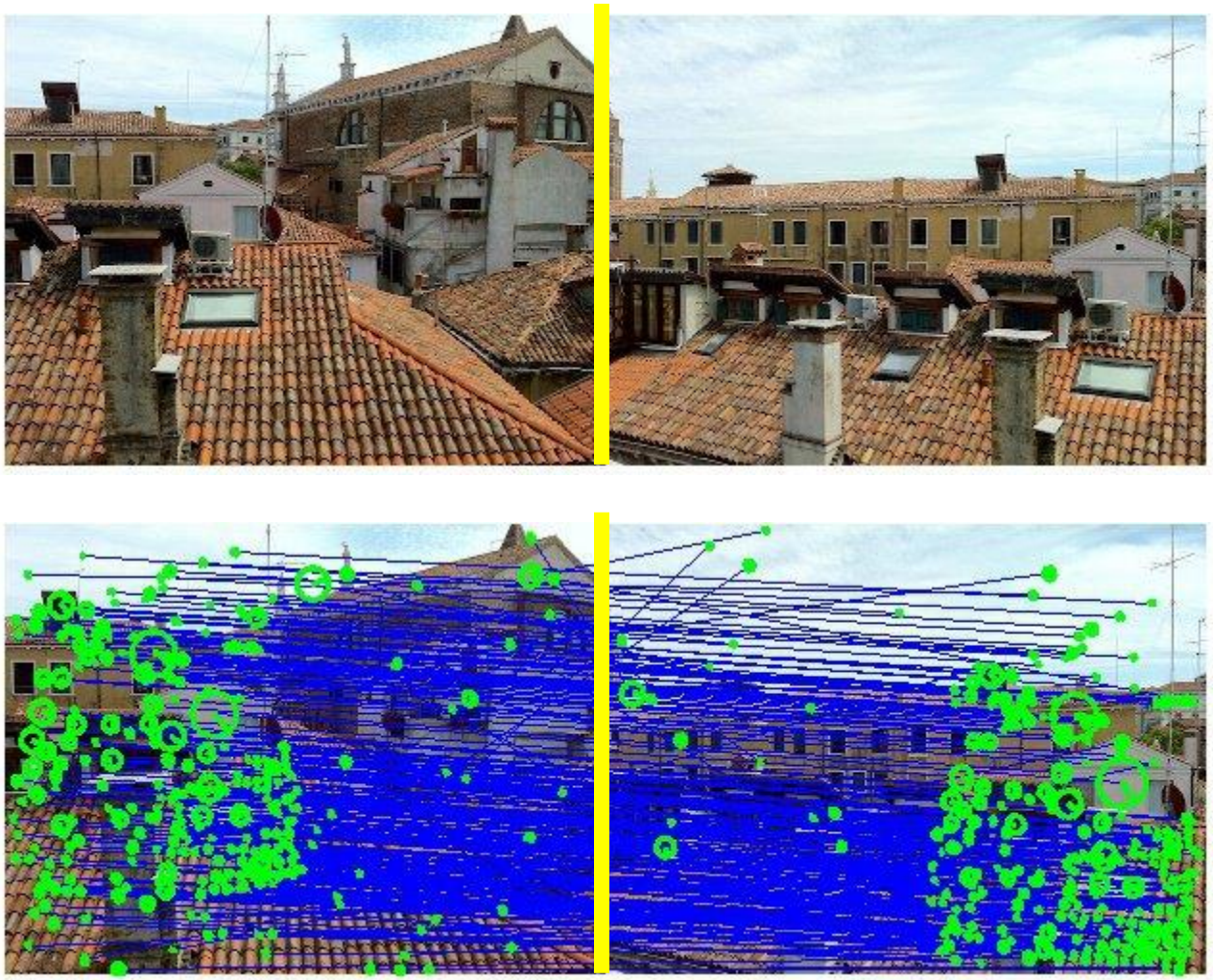
Interest points and
their scales and
orientations
(random subset of 50)



SIFT descriptors

http://www.vlfeat.org/overview/sift.html

Recall: SIFT (preliminary) matches

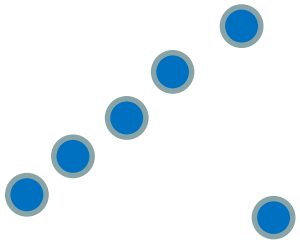


Not all of these are valid matches!

http://www.vlfeat.org/overview/sift.html

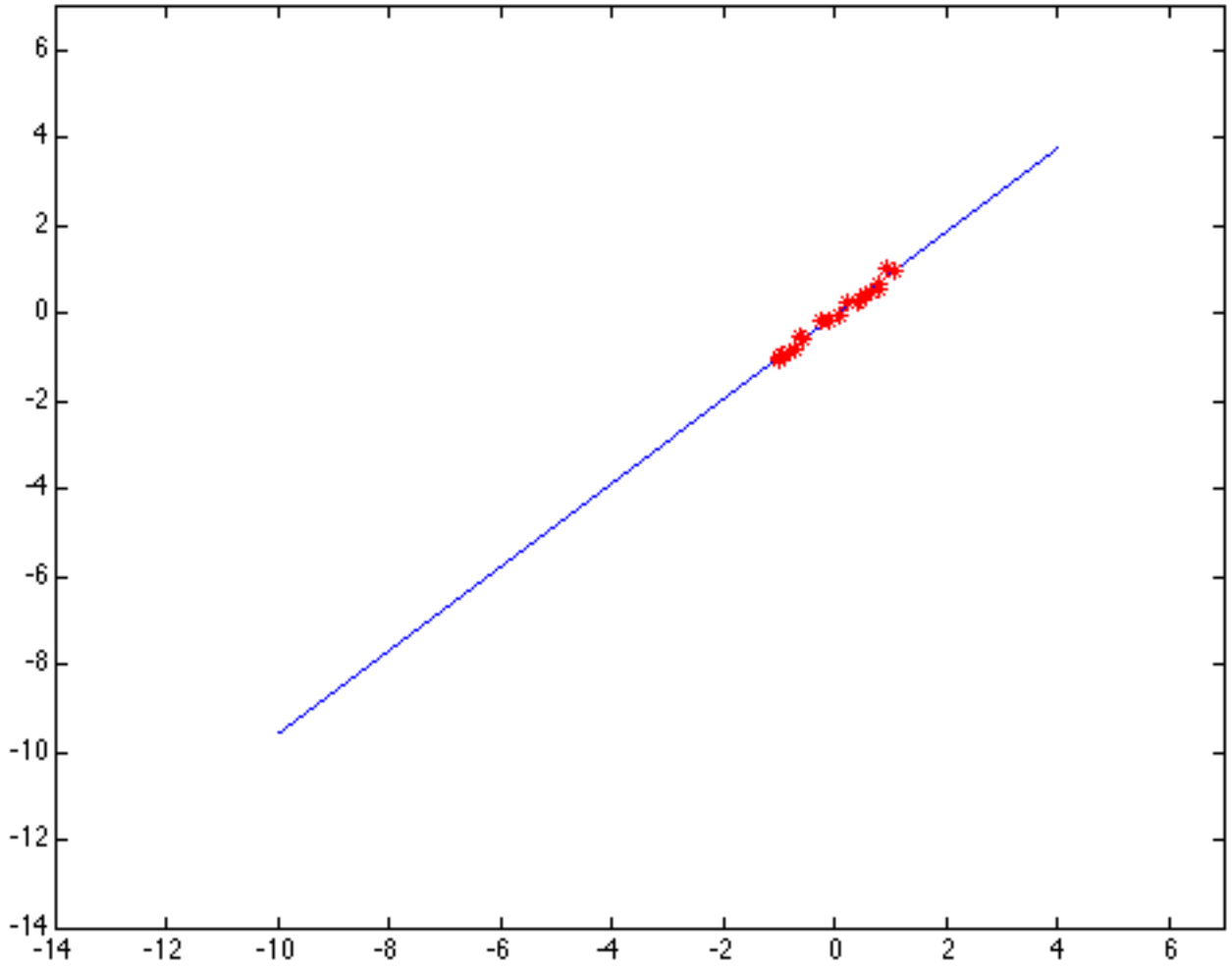
Outliers

- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn't belong to the line we are fitting.

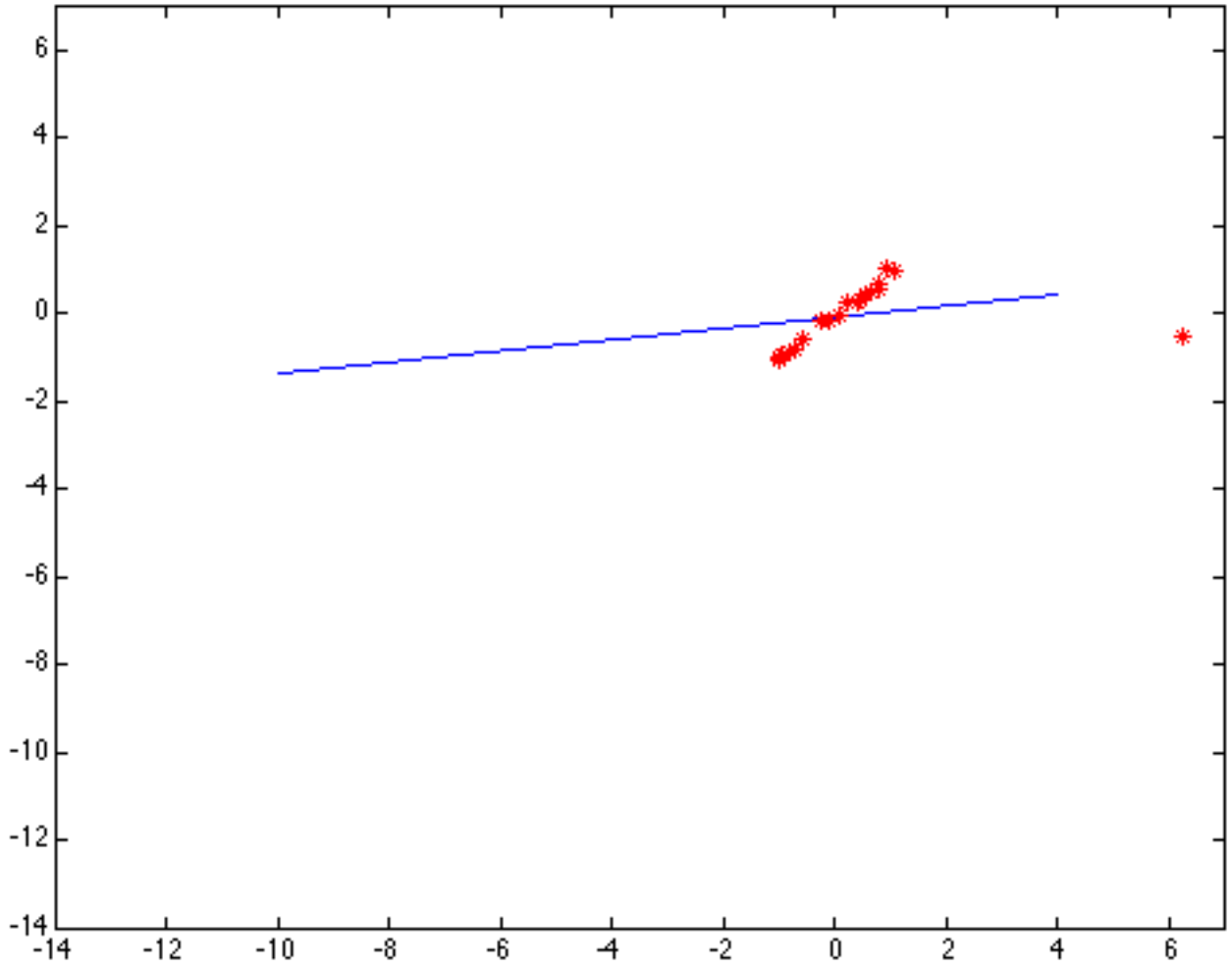


Kristen Grauman

Outliers affect least squares fit



Outliers affect least squares fit

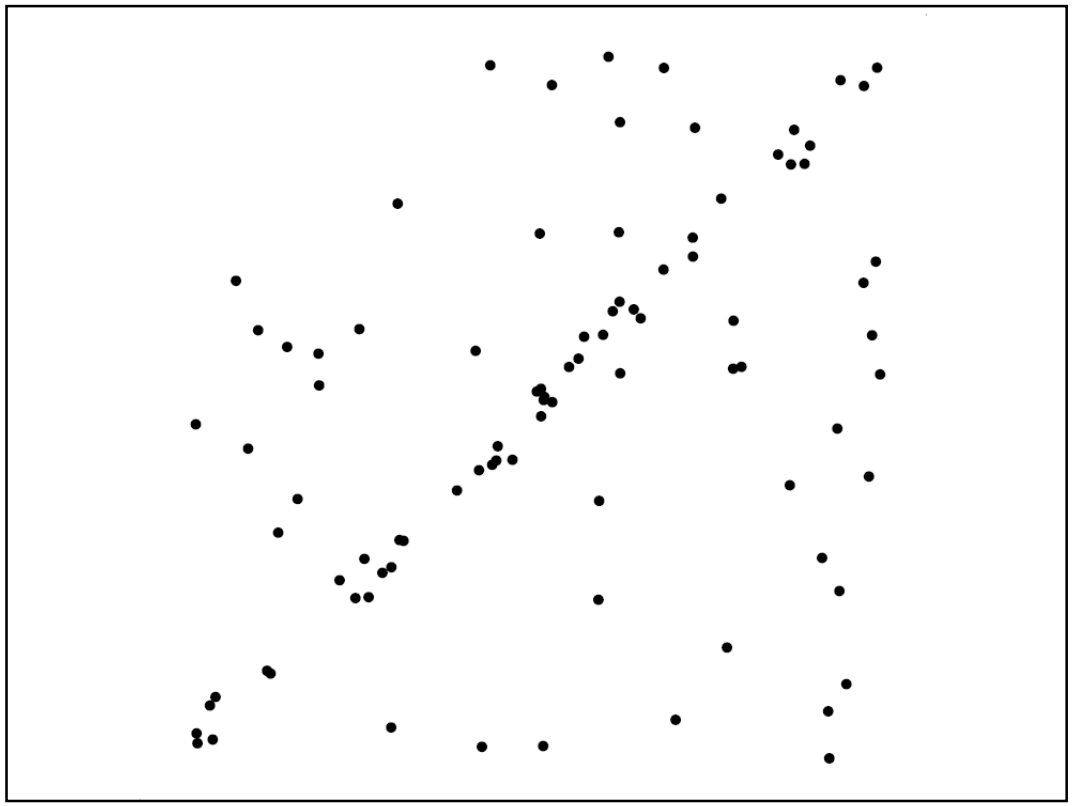


RANSAC

- RANdom SAmple Consensus (Fischler, Bolles 1981)
- **Approach**: we want to avoid the impact of outliers, so let's look for "inliers", and use those only.
- **Intuition**: if an outlier is chosen to compute the current fit, then the resulting line (transformation) won't have much support from rest of the points (matches).

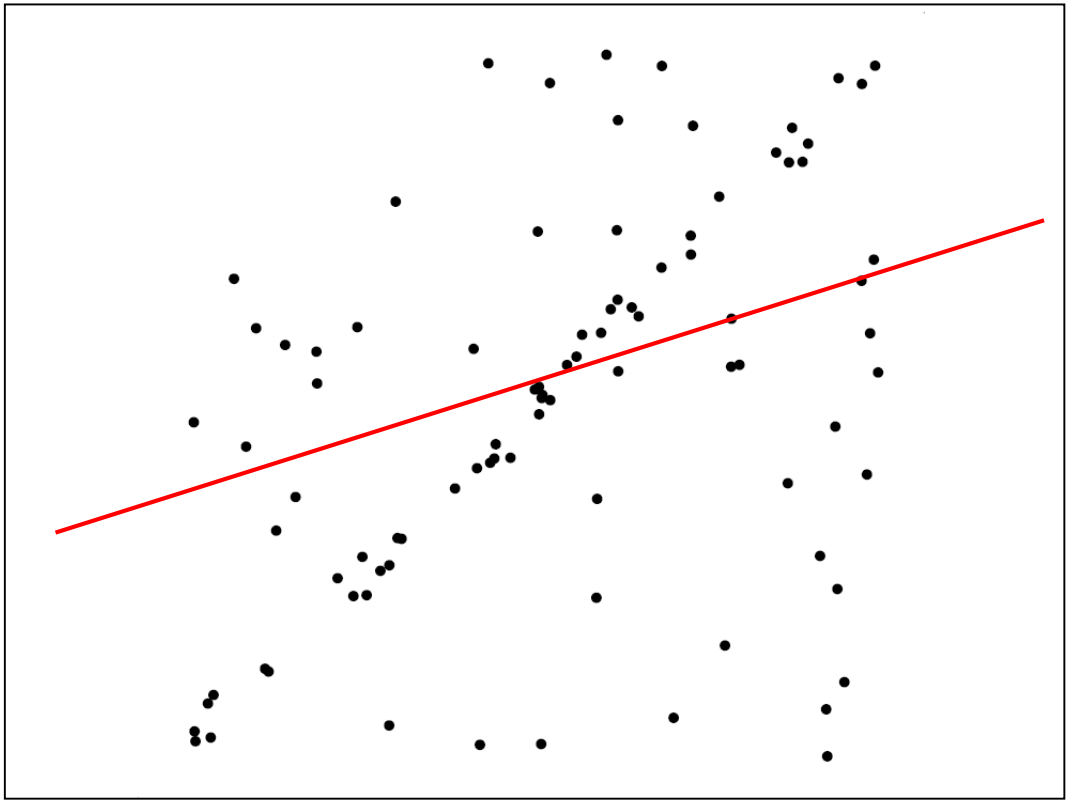
Kristen Grauman

RANSAC for line fitting example



Source: R. Raguram

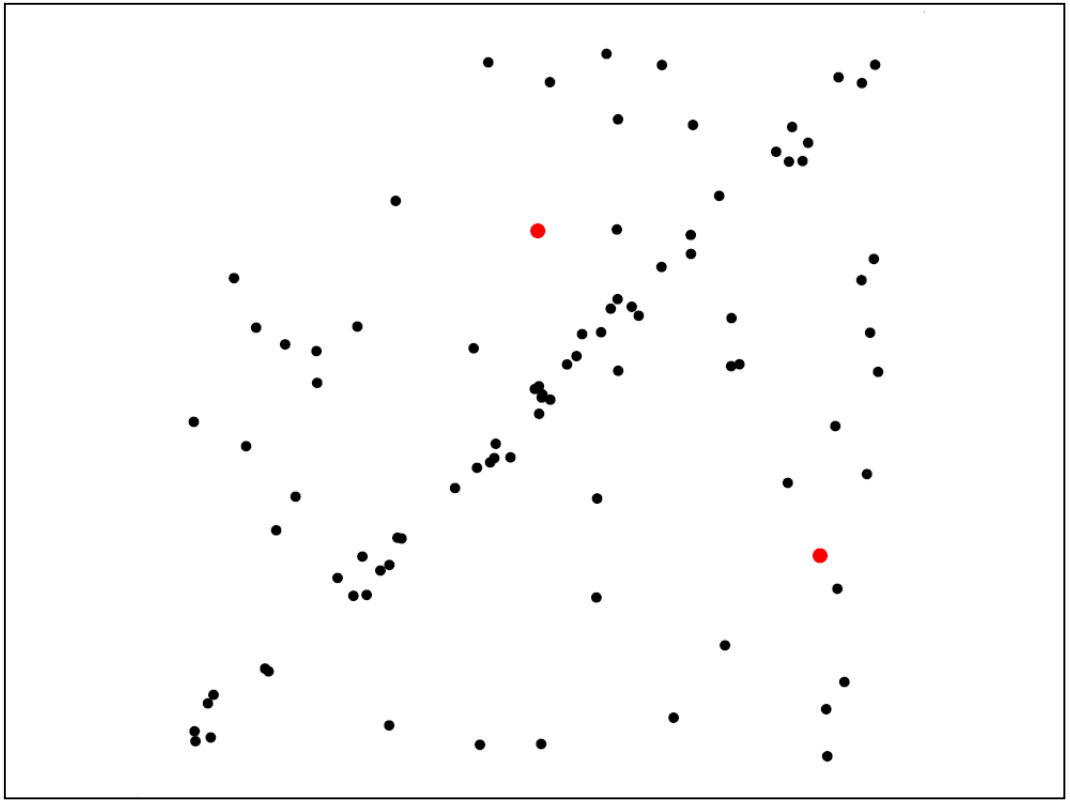
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

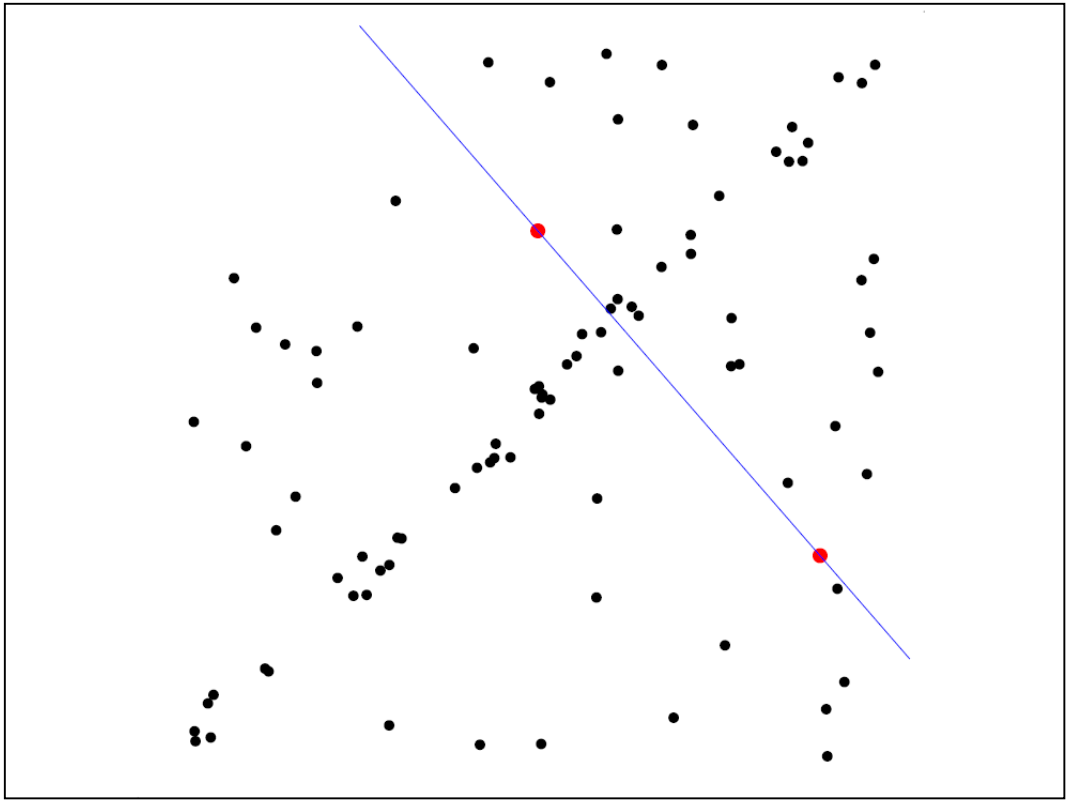
RANSAC for line fitting example



1. Randomly select *minimal* subset of points

Source: R. Raguram

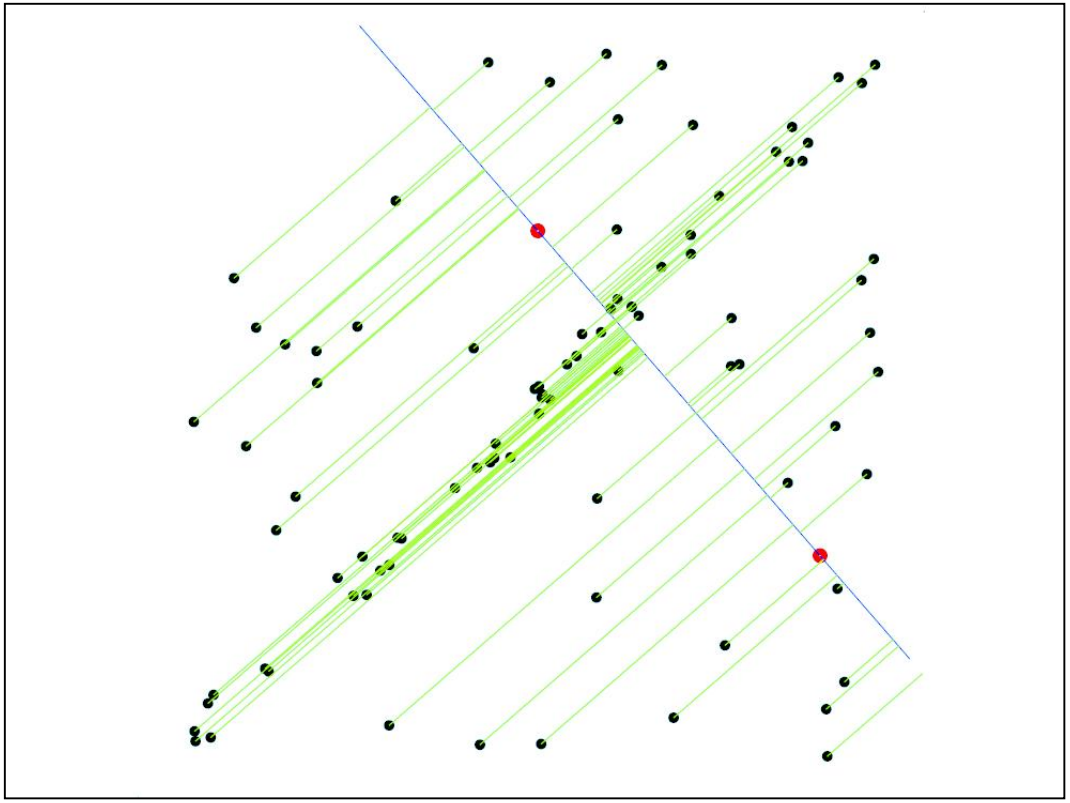
RANSAC for line fitting example



1. Randomly select *minimal* subset of points
2. Hypothesize a *model*

Source: R. Raguram

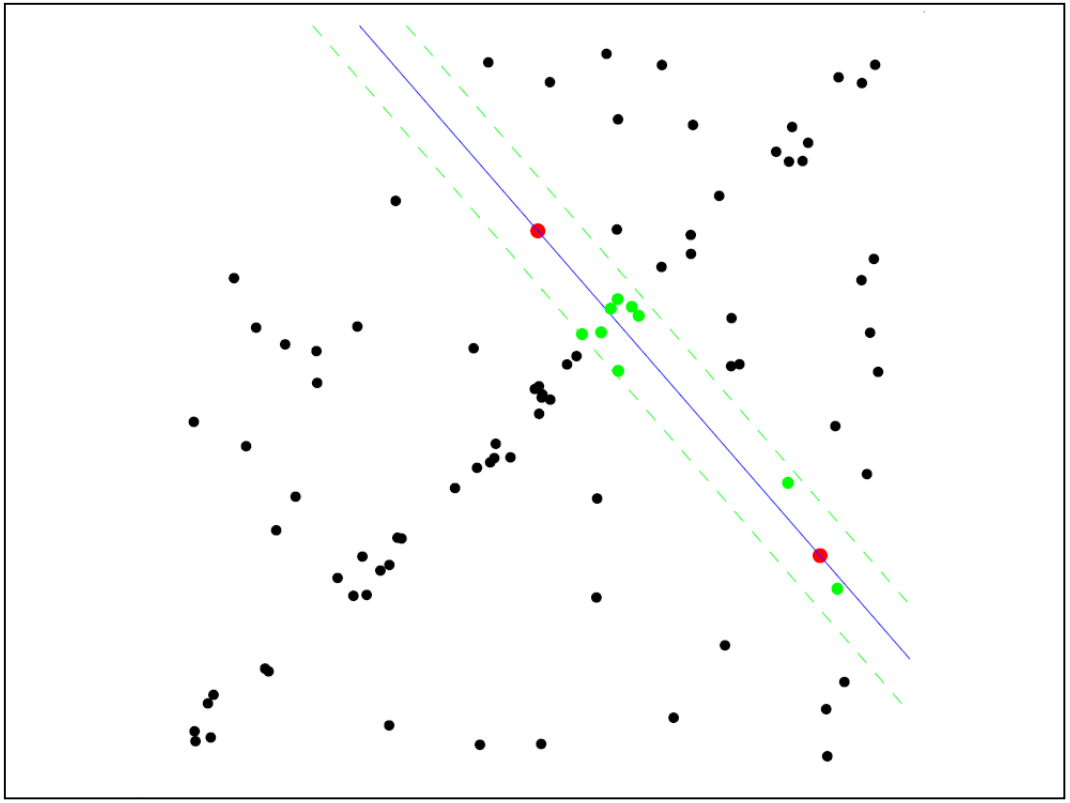
RANSAC for line fitting example



1. Randomly select *minimal* subset of points
2. Hypothesize a *model*
3. Compute *error* function

Source: R. Raguram

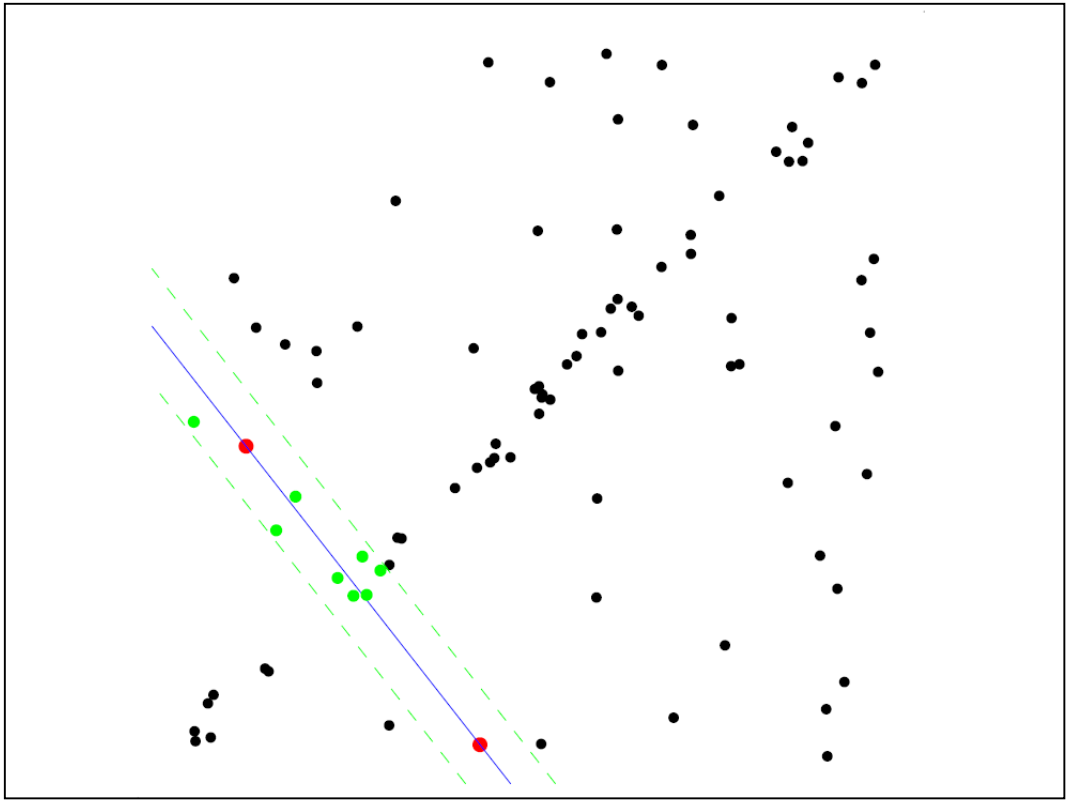
RANSAC for line fitting example



1. Randomly select *minimal* subset of points
2. Hypothesize a *model*
3. Compute *error* function
4. Select points consistent with model

Source: R. Raguram

RANSAC for line fitting example

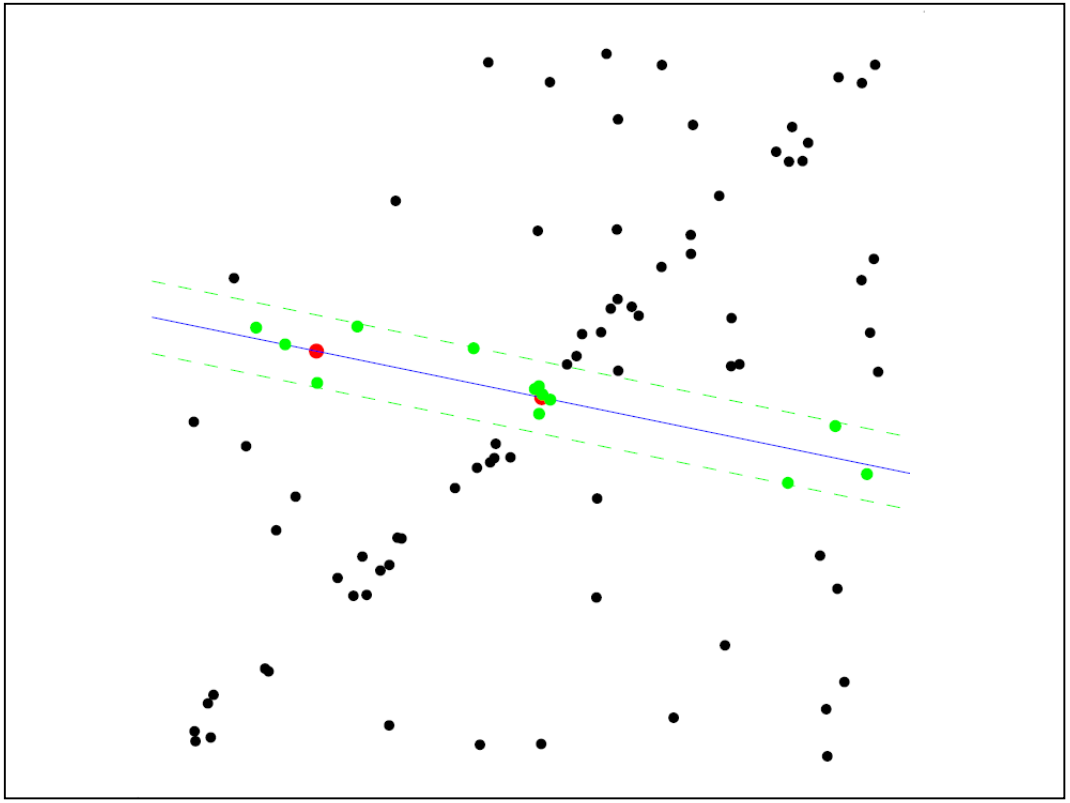


1. Randomly select *minimal* subset of points
2. Hypothesize a *model*
3. Compute *error* function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example



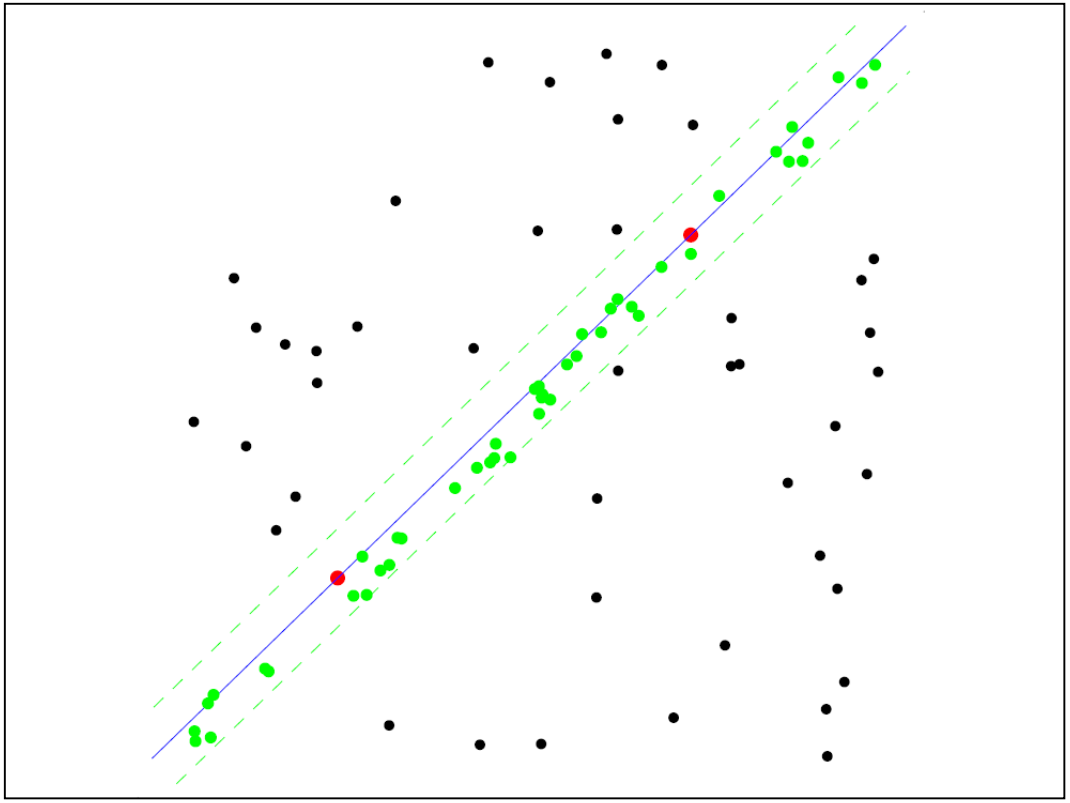
1. Randomly select *minimal* subset of points
2. Hypothesize a *model*
3. Compute *error* function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

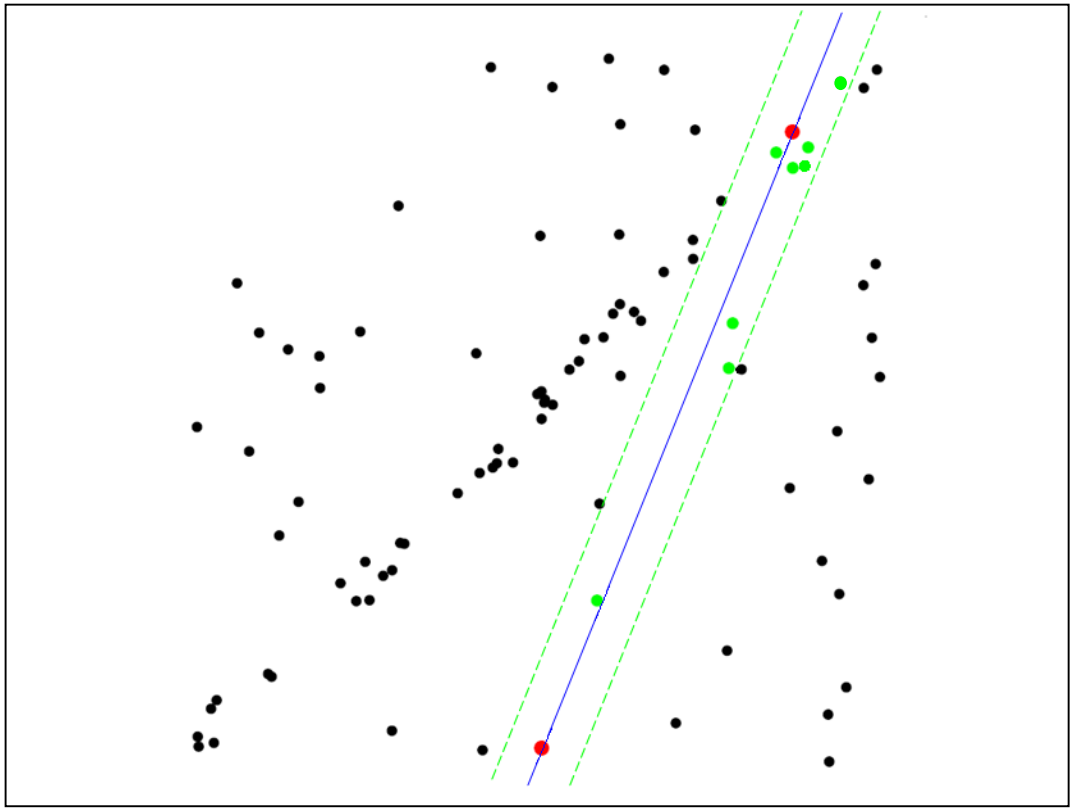
Uncontaminated sample



1. Randomly select *minimal* subset of points
2. Hypothesize a *model*
3. Compute *error* function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example



1. Randomly select *minimal* subset of points
2. Hypothesize a *model*
3. Compute *error* function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting

Repeat (trials):

- Draw ***k*** points uniformly at random
- Fit line to these ***k*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- Stopping criterion - accept the line:
  - option 1: if there are ***d*** or more inliers
  - option 2: after ***n*** trials

Refit line using all inliers

How many trials for RANSAC?

To ensure good chance of finding true inliers, need sufficient number of trials, S.

- Let P be to the total prob. of success after S trials.
- Let p be prob. that a given match is valid (*inlier rate*)
- Prob. that all **k** random samples in one trial are inliers: p^k
- Prob. that all S trials will fail:

$$1-P = (1-p^k)^S$$

- Required minimum number of trials:

$$S = \log(1-P) \;/\; \log(1-p^k)$$

for P = 0.99:

k	p	S
3	0.5	35
6	0.6	97
6	0.5	293
7	0.2	359776
7	0.1	46051700

RANSAC for line fitting

Repeat (trials):

- Draw ***k*** points uniformly at random
- Fit line to these ***k*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- Stopping criterion - accept the line:
  - option 1: if there are ***d*** or more inliers
  - option 2: after ***n*** trials
  - option 3 (dynamic): after ~ log(1-P) / log(1-p^k) trials
    - » where p is the best found inlier rate

Refit using all inliers

That is an example fitting a *model*

(line)…

What about fitting a *transformation*

(translation)?

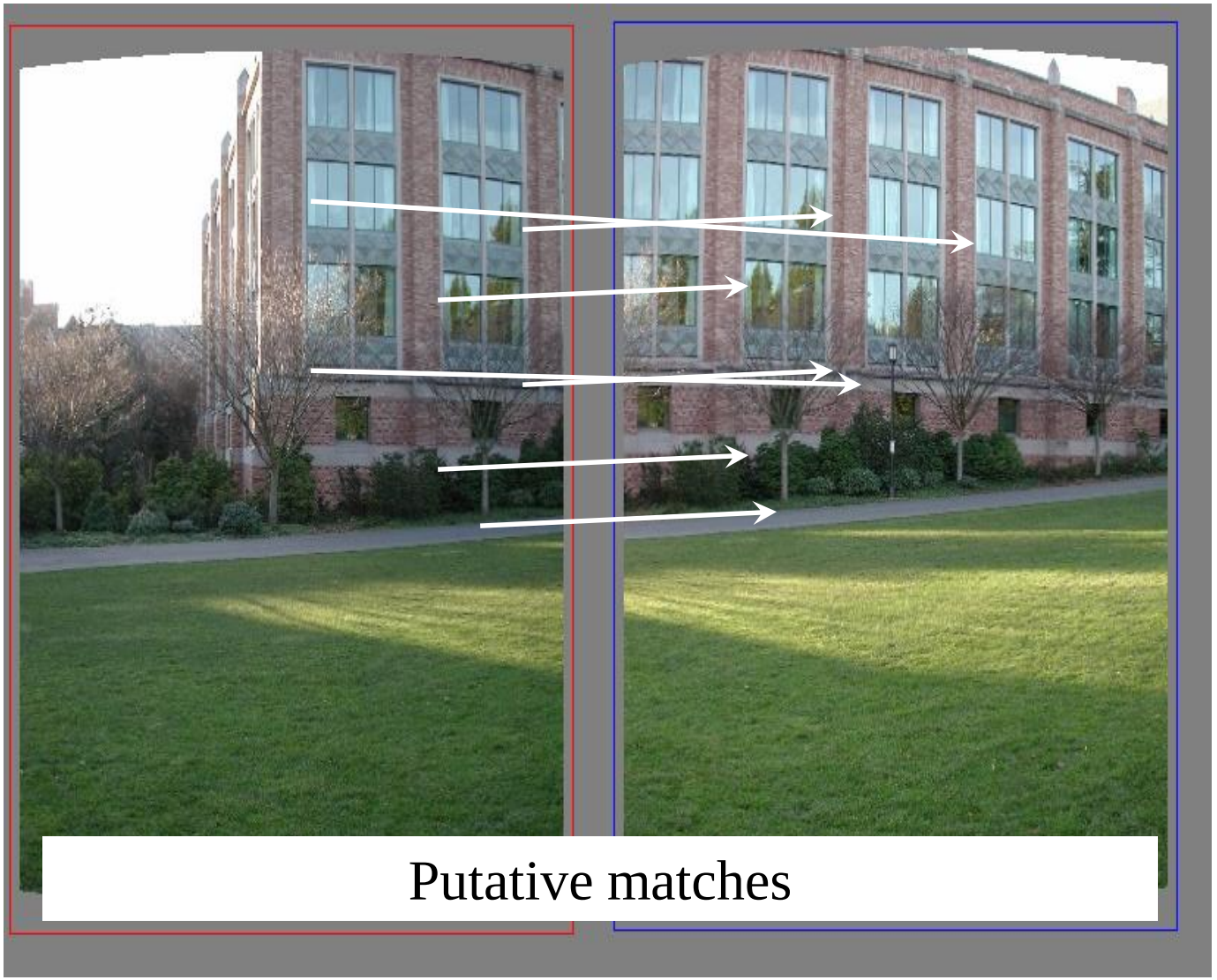
RANSAC: General form

- RANSAC loop:

1. Randomly select a *seed group* on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute estimate of transformation on all of the inliers

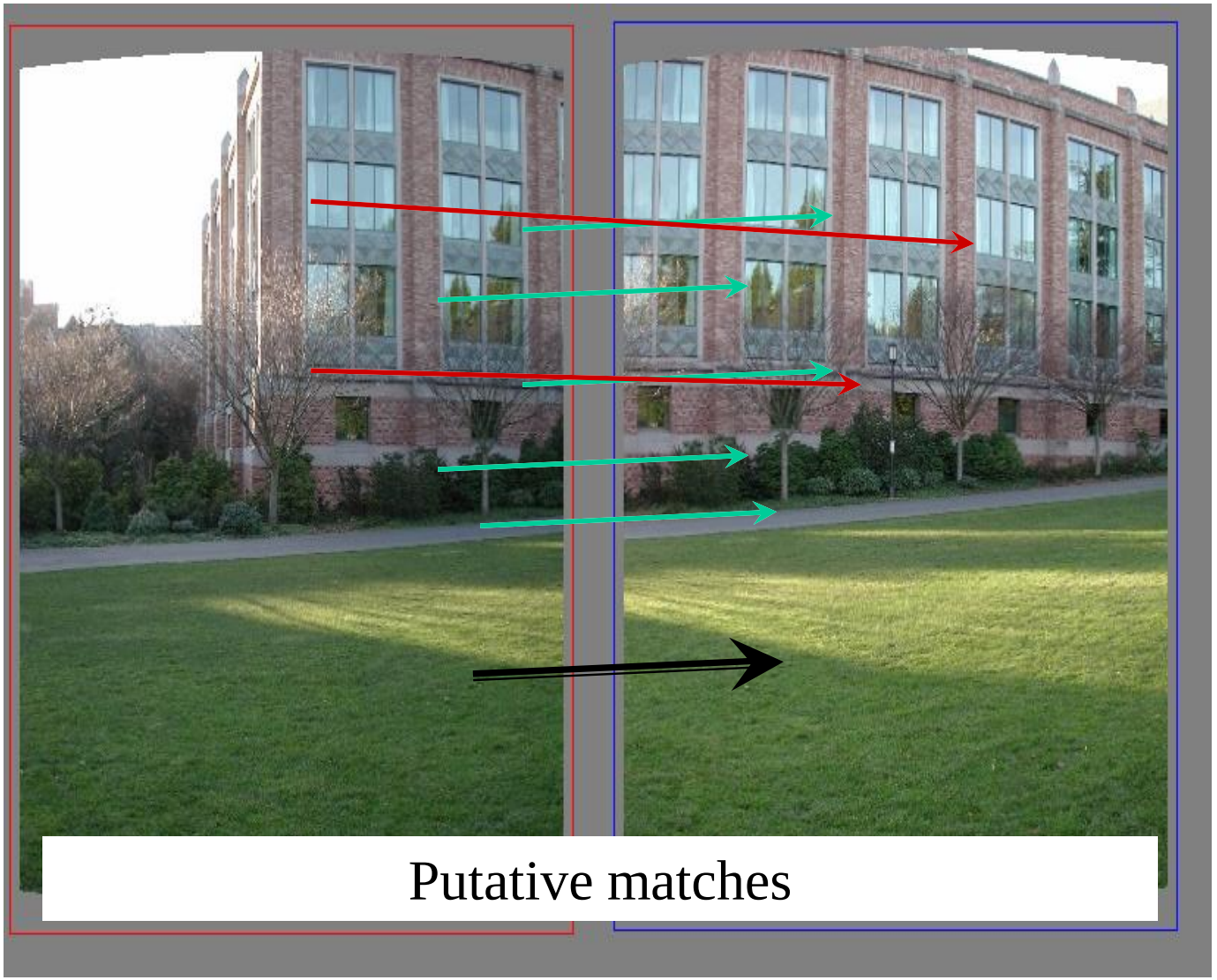
- Keep (and refine) the transformation with the largest number of inliers

RANSAC example: Translation



Source: Rick Szeliski

RANSAC example: Translation



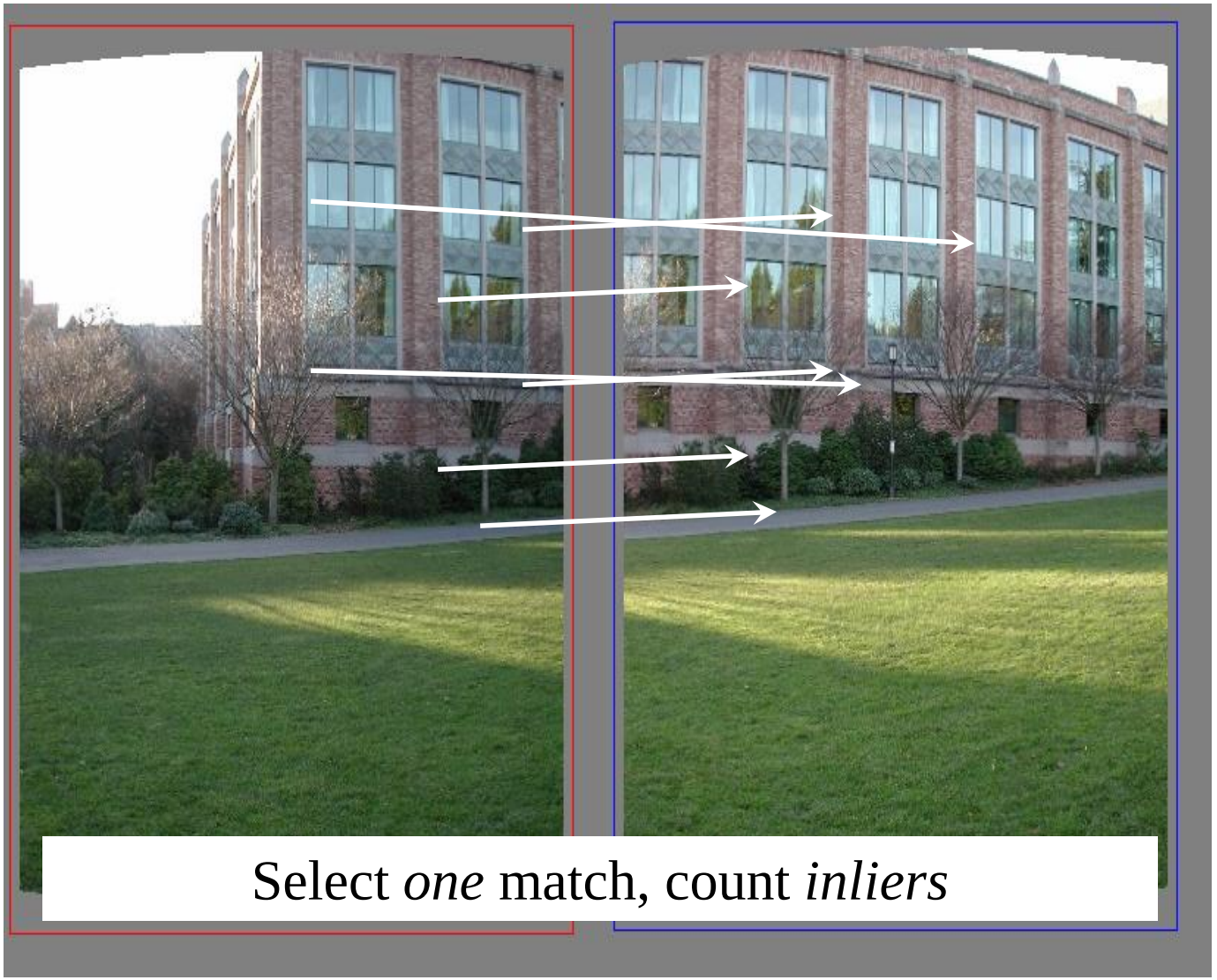
Source: Rick Szeliski

RANSAC example: Translation

seed match

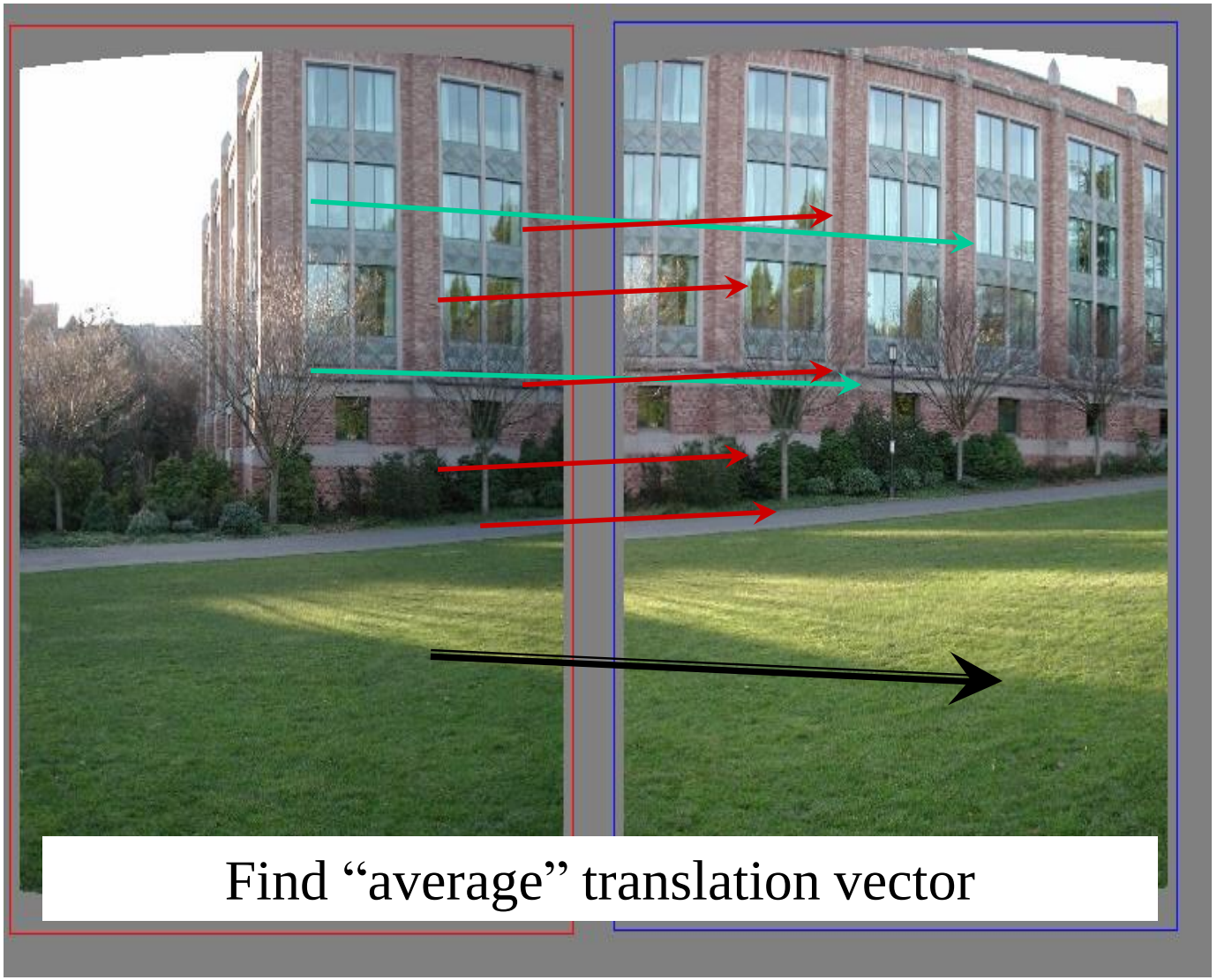
model inliers

model outliers



Source: Rick Szeliski

RANSAC example: Translation



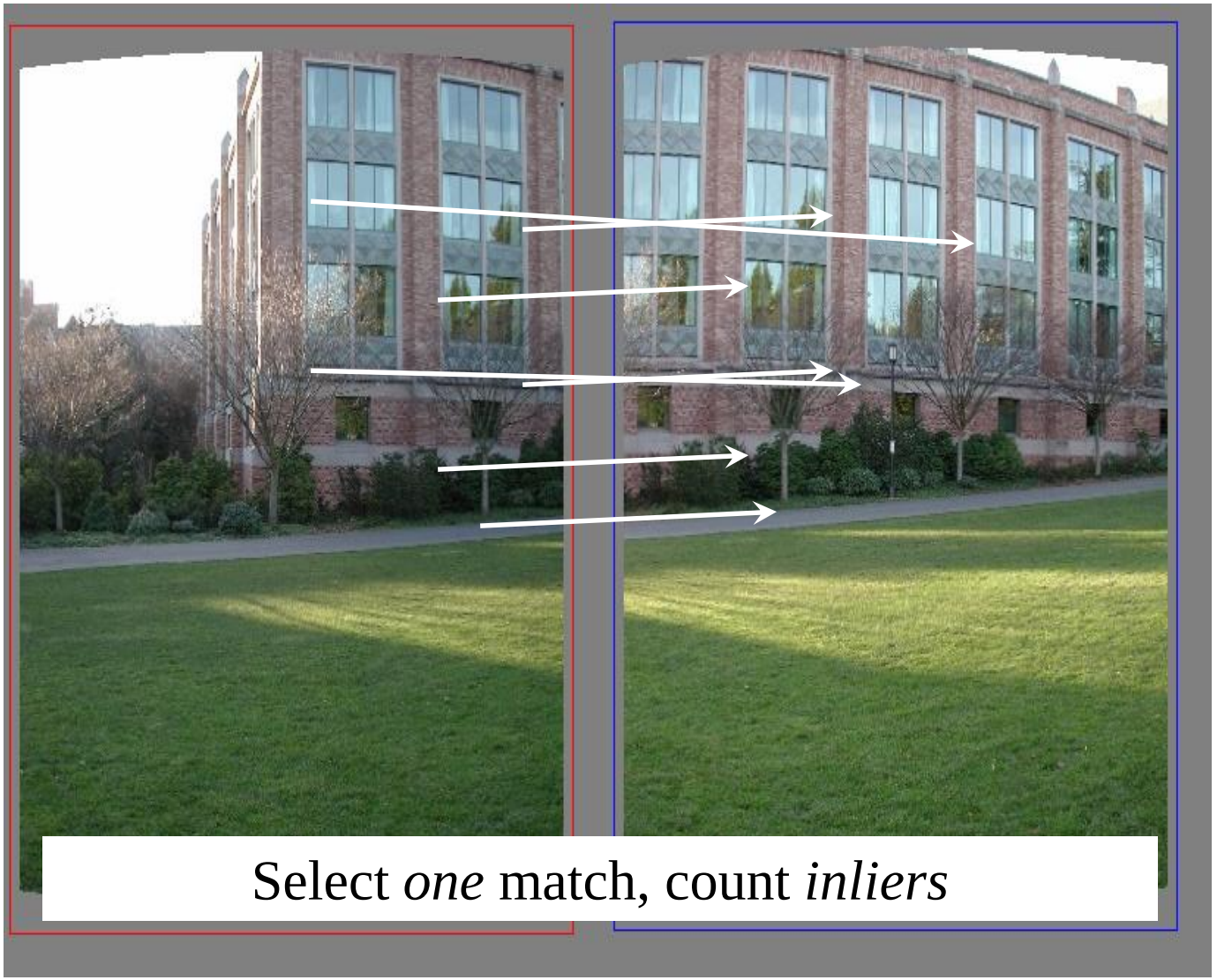
Source: Rick Szeliski

RANSAC example: Translation

seed match

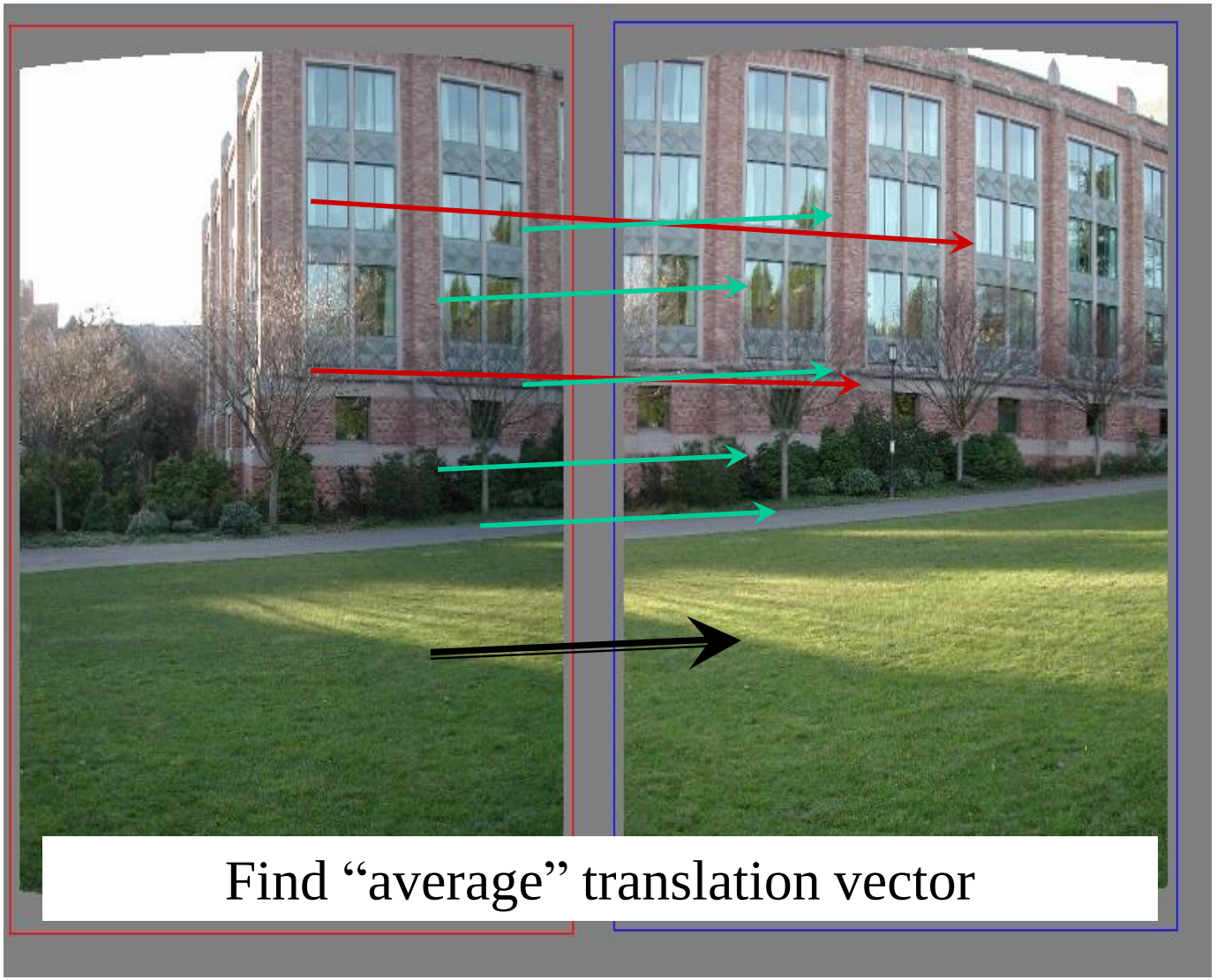
model inliers

model outliers



Source: Rick Szeliski

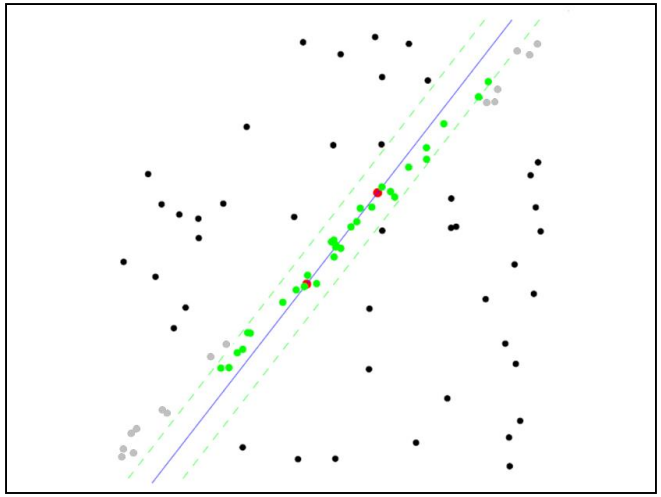
RANSAC example: Translation



Source: Rick Szeliski

RANSAC pros and cons

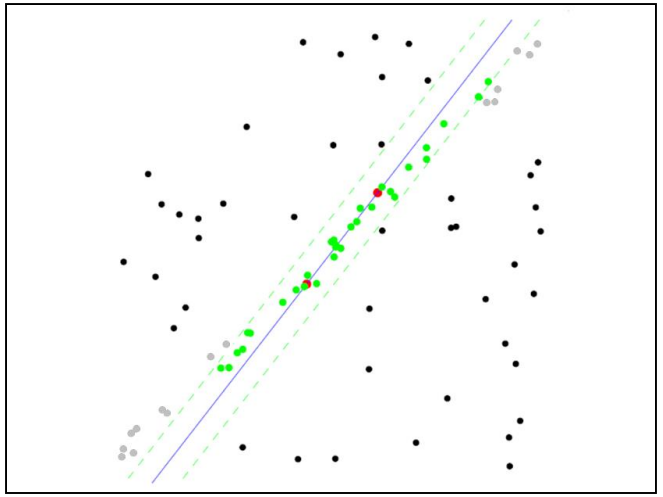
- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Parameters to tune
  - Doesn’t work well for low inlier ratios (too many iterations, or can fail completely)
  - Can’t always get a good initialization of the model based on the minimum number of samples



Slide credit: Lana Lazebnik

RANSAC popular extensions

- Pro-SAC (speed)
  - Rank matches by their quality and sample with non-uniform distribution.
- LO-SAC (precision)
  - Perform local optimization on inliers of each promising model
- MLE-SAC (robustness)
  - Maximize the likelihood rather than the number of inliers
- many others…



Recap

- Feature-based alignment
  - 2D transformations
  - model fitting
  - RANSAC