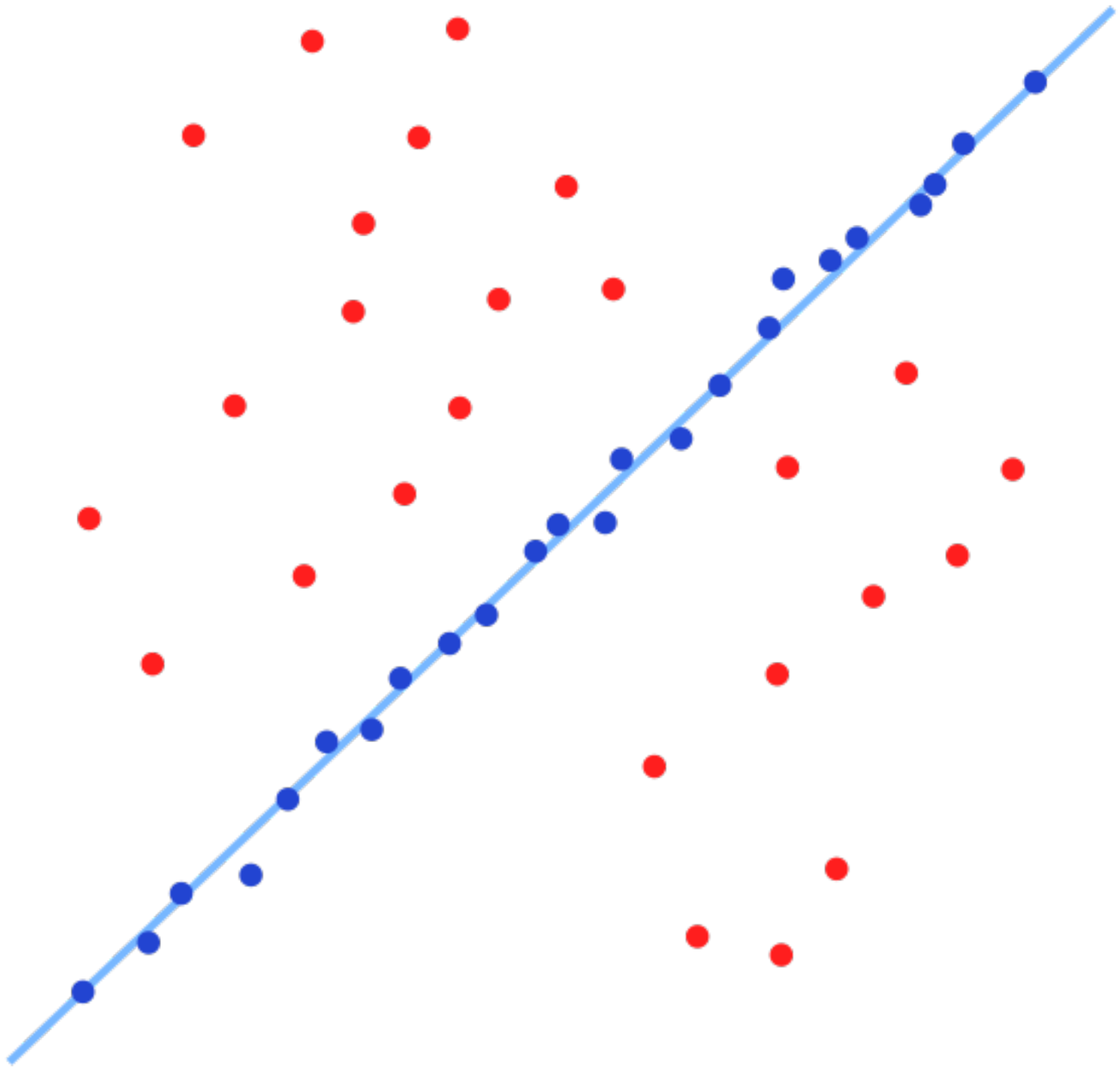
3D Reconstruction with Computer Vision
Meeting 3: Fitting a model



CS 378 Fall 2014, UT Austin, Bryan Klingner, 4 September

Slides mostly stolen from Alyosha Efros and Kristen Grauman

Alignment problem

- In alignment, we will fit the parameters of some **transformation** according to a set of matching feature pairs ("correspondences").

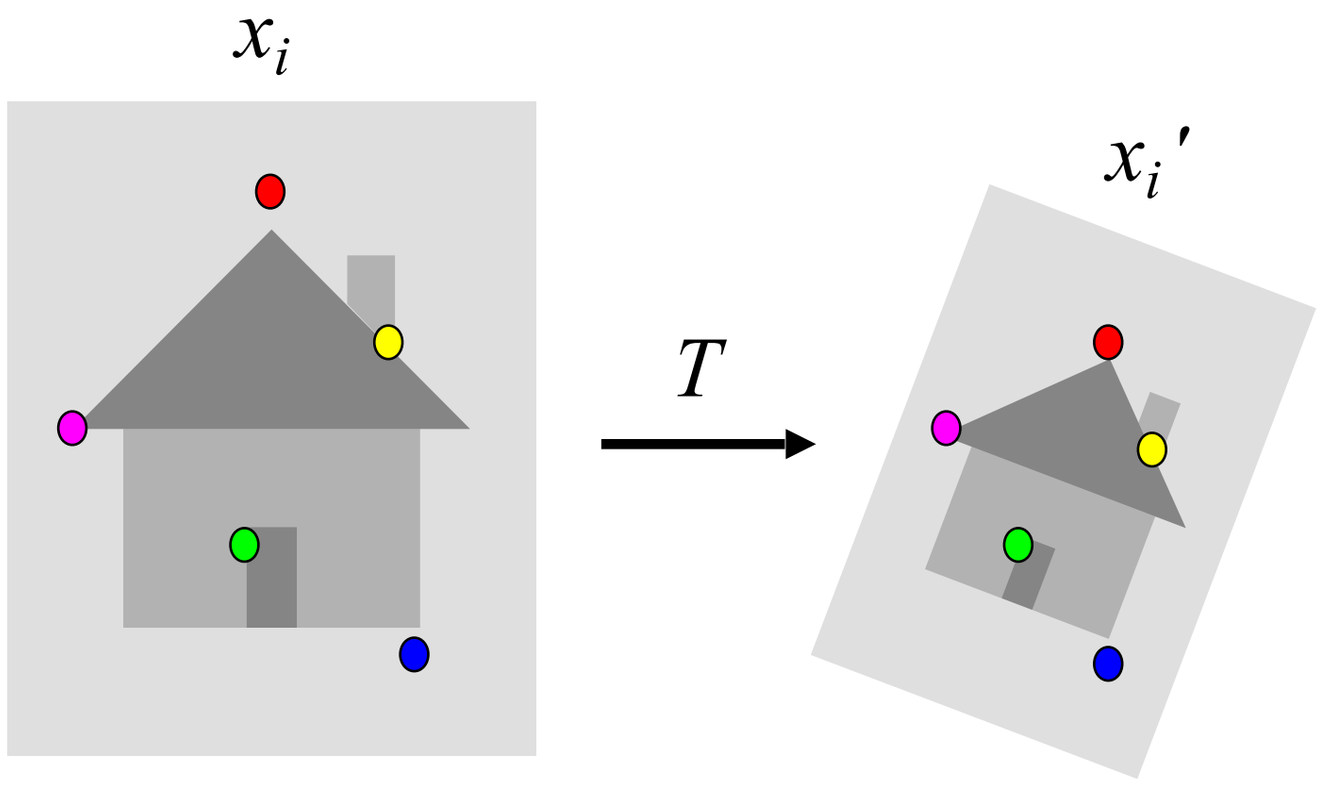
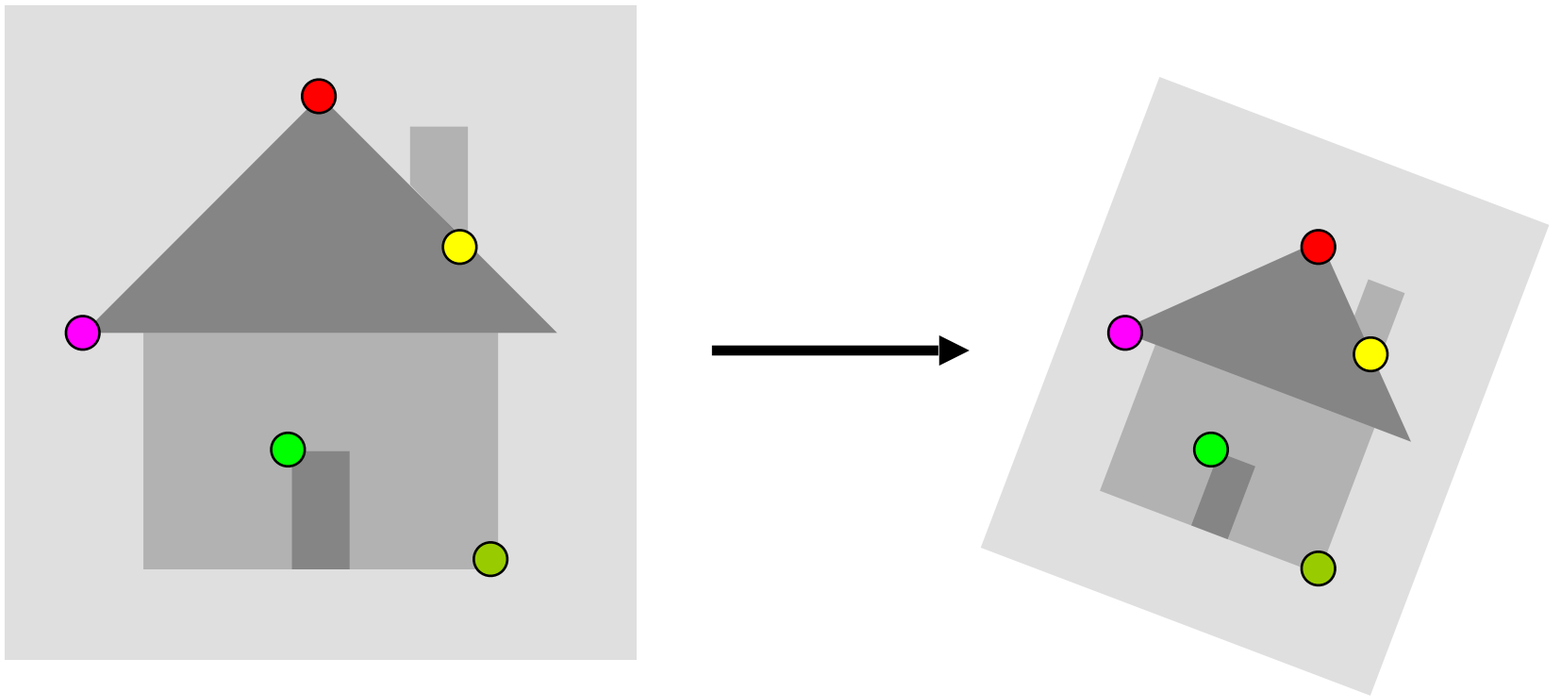


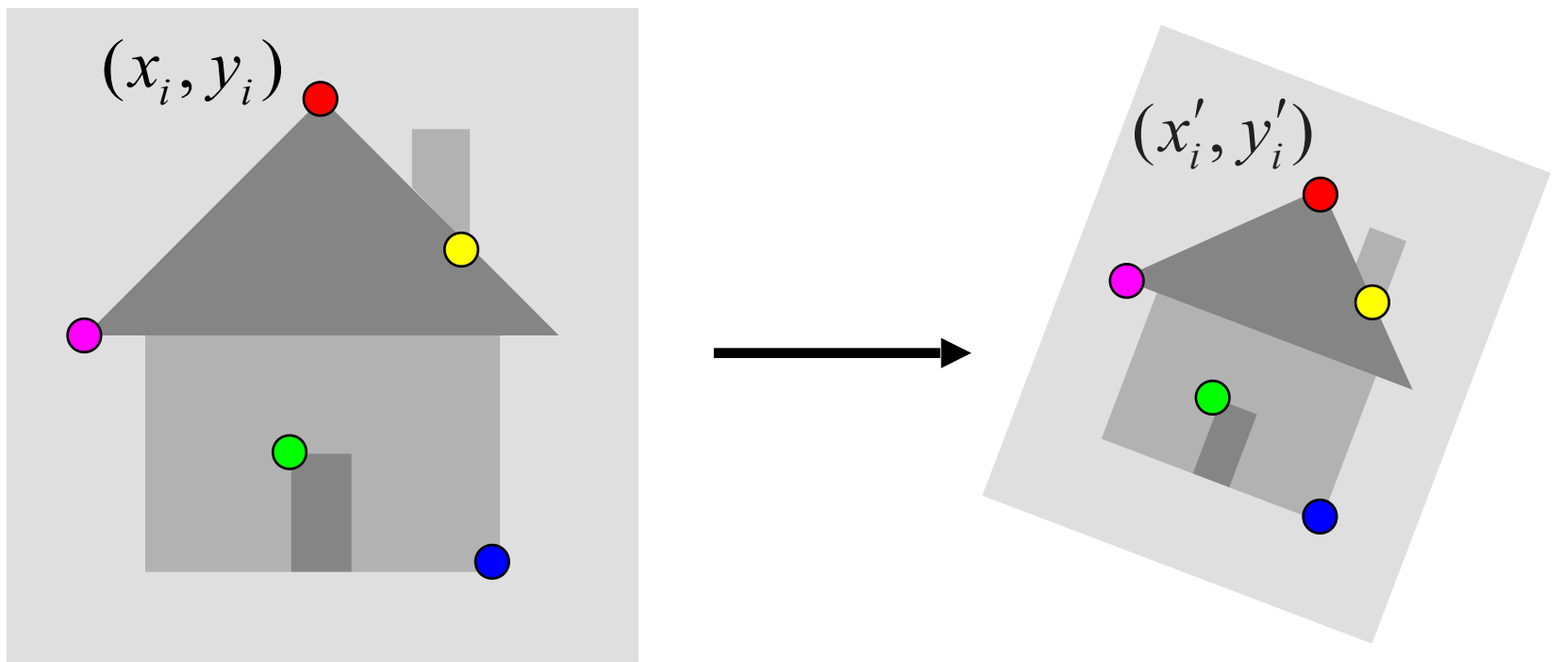
Image alignment



- Two broad approaches:
  - Direct (pixel-based) alignment
    - Search for alignment where most pixels agree
  - Feature-based alignment
    - Search for alignment where ***extracted features*** agree
    - Can be verified using pixel-based alignment

Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?



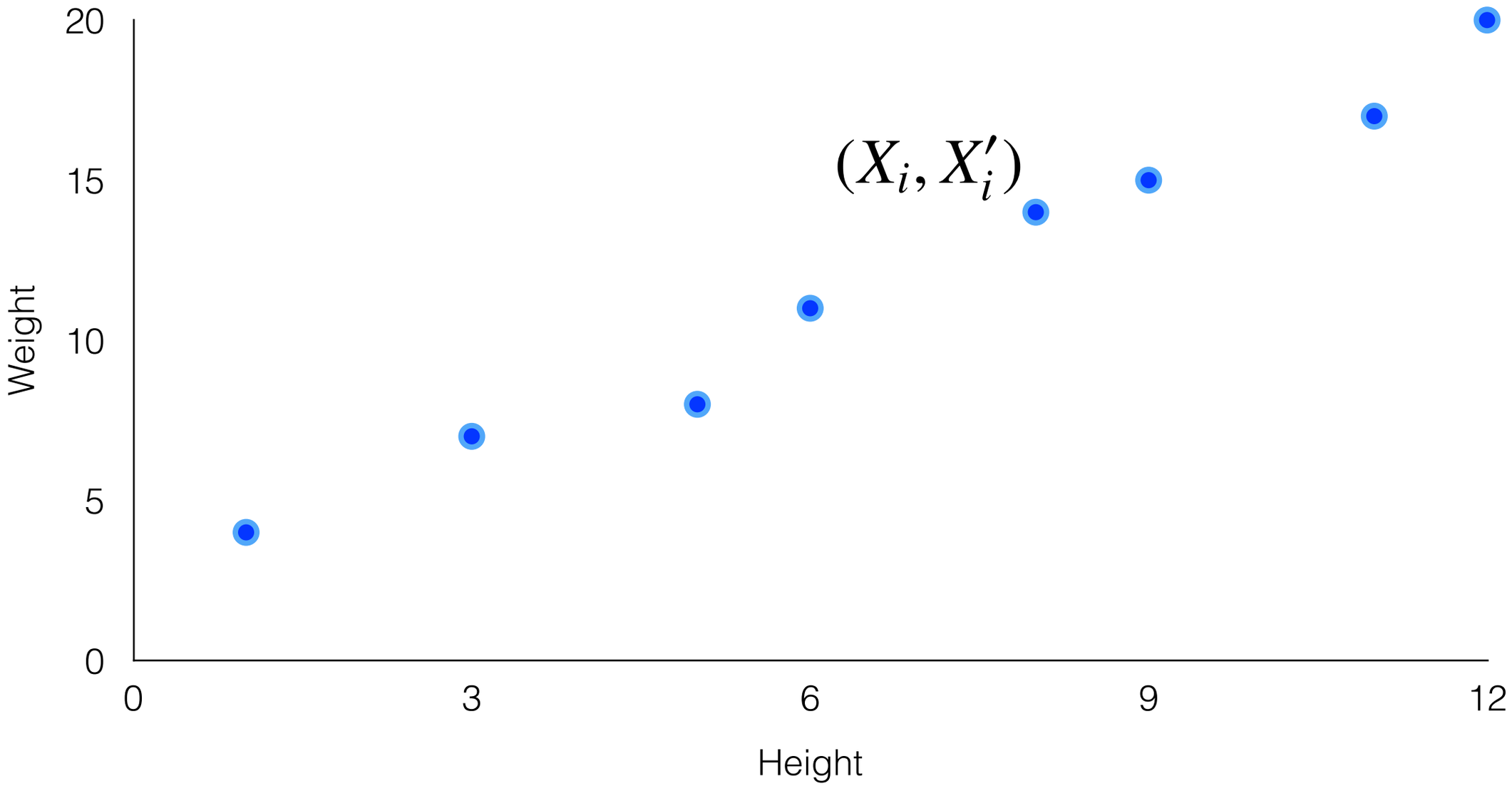
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Least-squares best fit

Say we have a set of data points:

$$(X_1, X_1'), (X_2, X_2'), (X_3, X_3'), ..., (X_n, X_n')$$

Where X_i is a person's height and X_i' is their weight

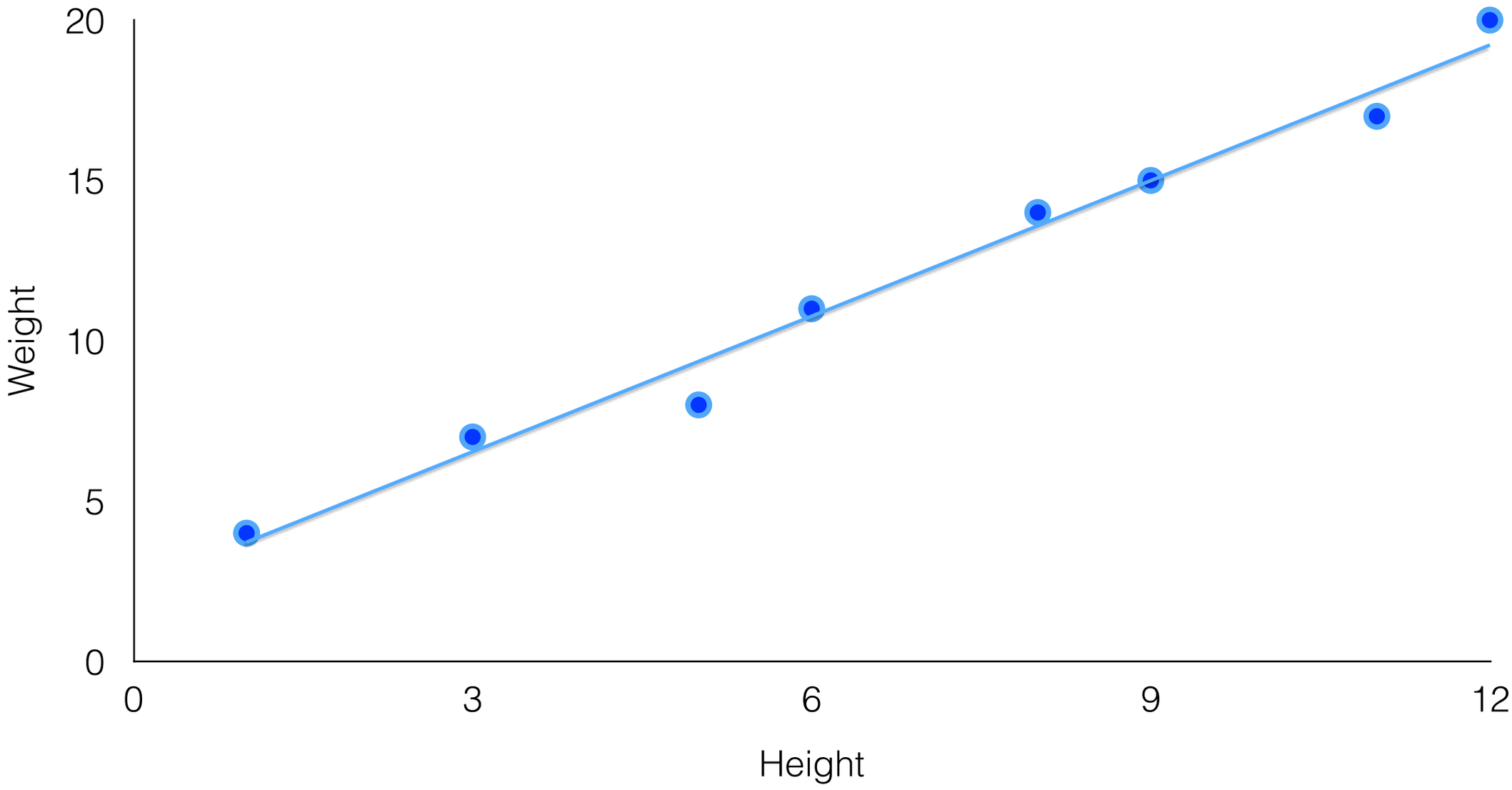


Least-squares best fit

We want to fit a **model** to predict weight from height.

Let's use a line: $Xa + b = X'$

We need to find **model parameters** a **and** b

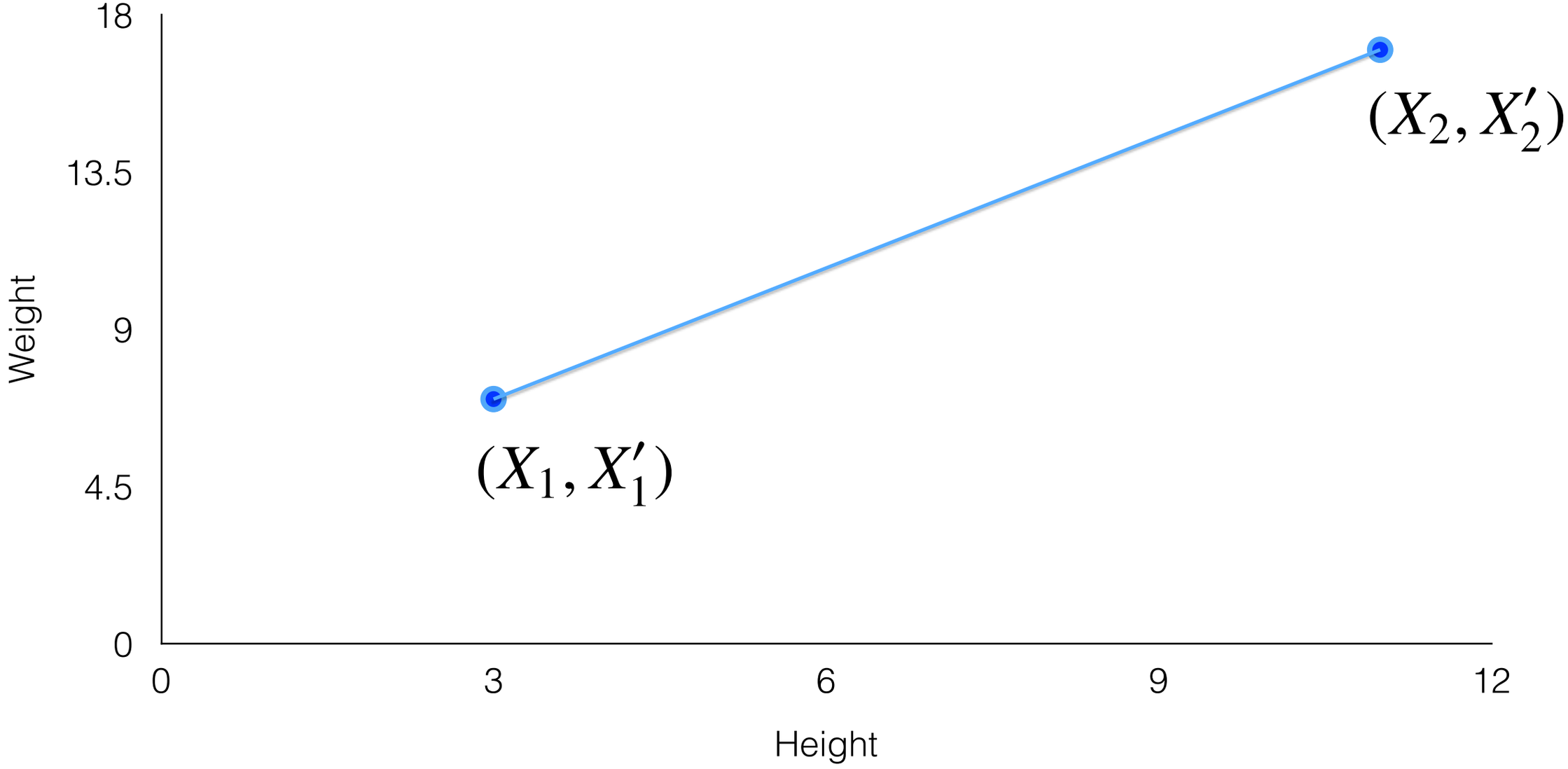


Least-squares best fit

How many data points do we need to resolve these two unknowns?

$$X_1 a + b = X_1'$$
$$X_2 a + b = X_2'$$

$$\begin{bmatrix} X_1 & 1 \\ X_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} X_1' \\ X_2' \end{bmatrix}$$



Least-squares best fit

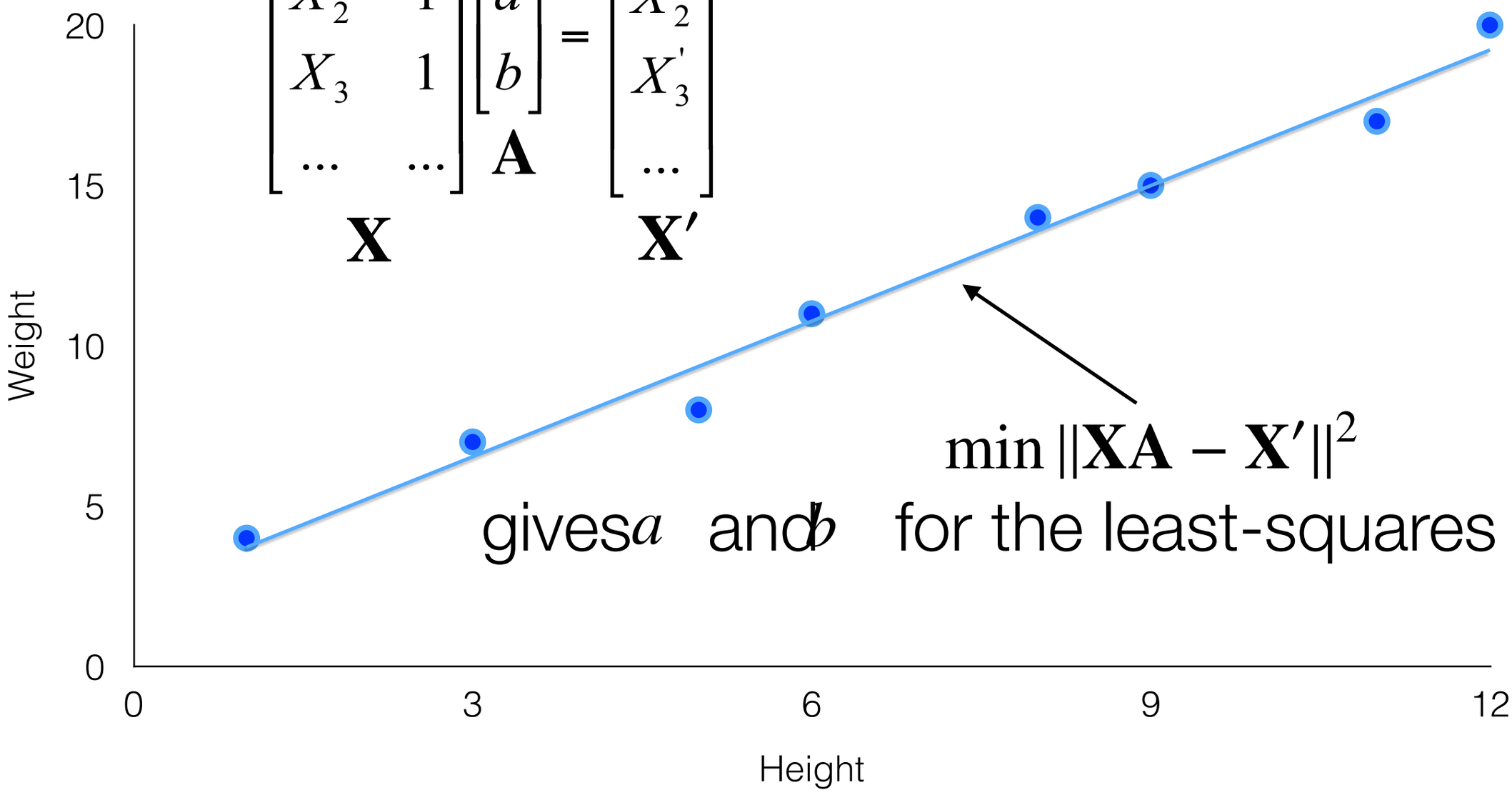
Add more points, and the system is **overconstrained.**

No one line passes through all the points.

So, we choose the one with the least total (squared) erro

$$\begin{bmatrix} X_1 & 1 \\ X_2 & 1 \\ X_3 & 1 \\ \dots & \dots \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} X_1' \\ X_2' \\ X_3' \\ \dots \end{bmatrix}$$

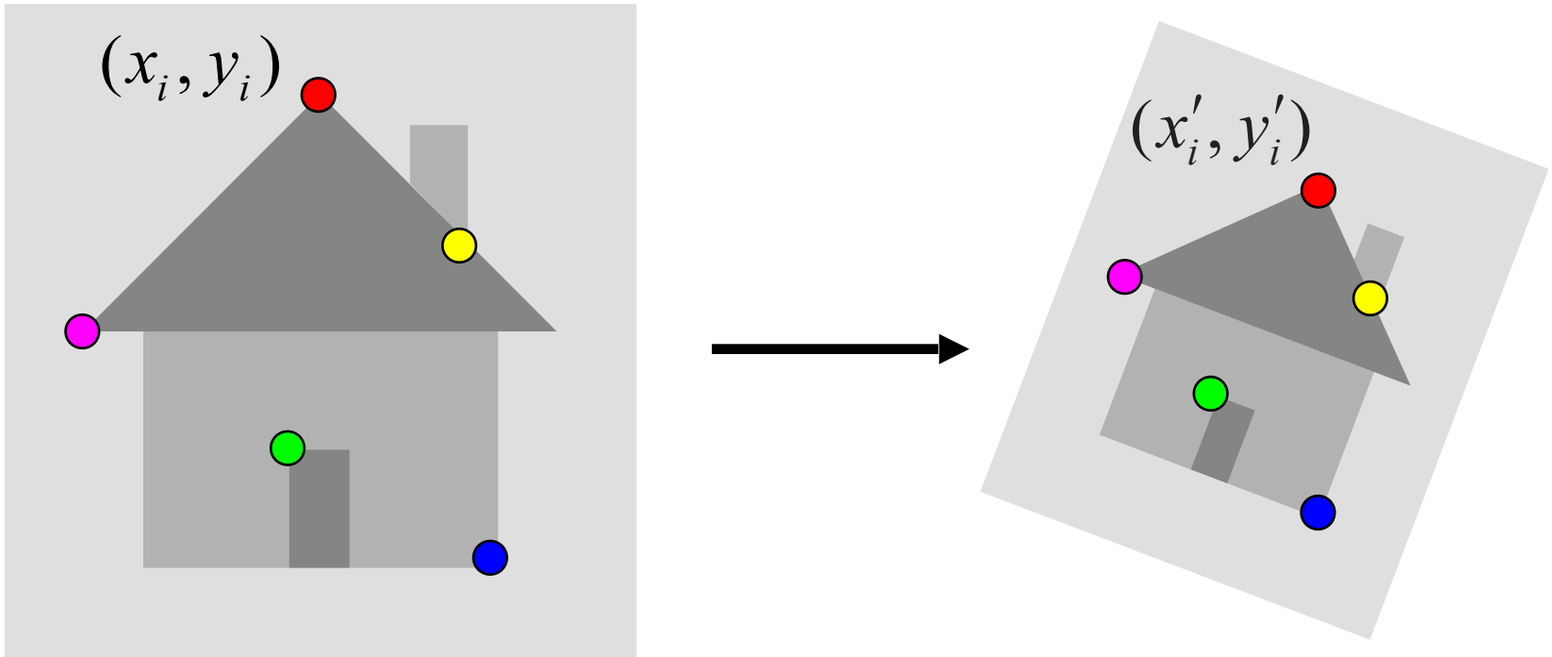
$$\mathbf{X} \qquad \mathbf{A} \qquad \mathbf{X}'$$



gives a and b for the least-squares best-fit l

Fitting a transformation (affine)

- Assuming we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & \\ & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

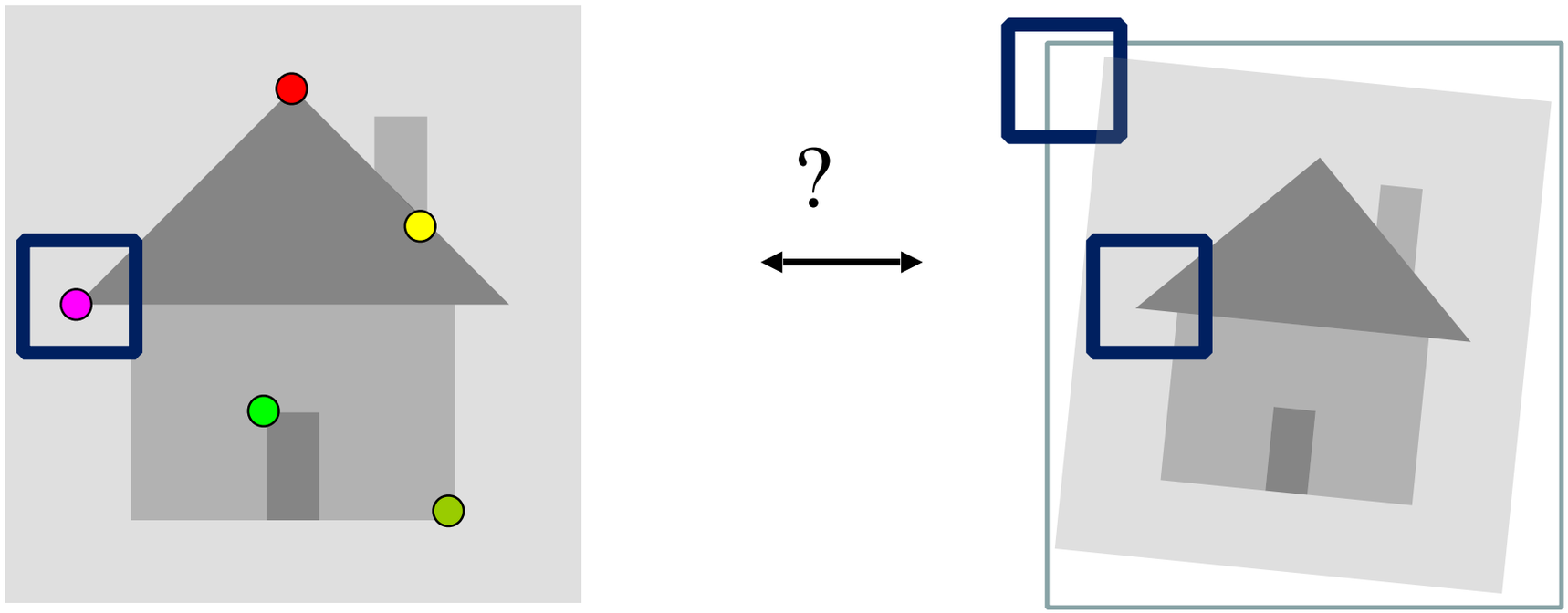
Fitting an affine transformation

$$\begin{bmatrix} & & \dots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \dots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \dots \\ x_i' \\ y_i' \\ \dots \end{bmatrix}$$

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?
- Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for ?
$$(x_{new}, y_{new})$$
- Where do the matches come from?

Kristen Grauman

What **are** the correspondences?



- Compare content in **local** patches, find best matches.

  e.g., simplest approach: scan with template, and compute SSD or correlation between list of pixel intensities in the patch

- Later in the course: how to select regions according to the geometric changes, and more robust descriptors.

Kristen Grauman

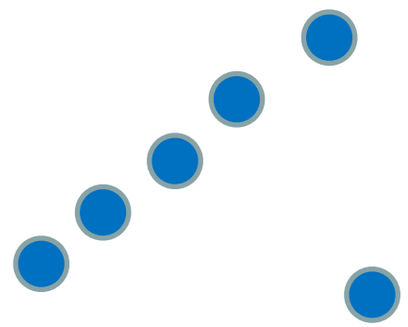
Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

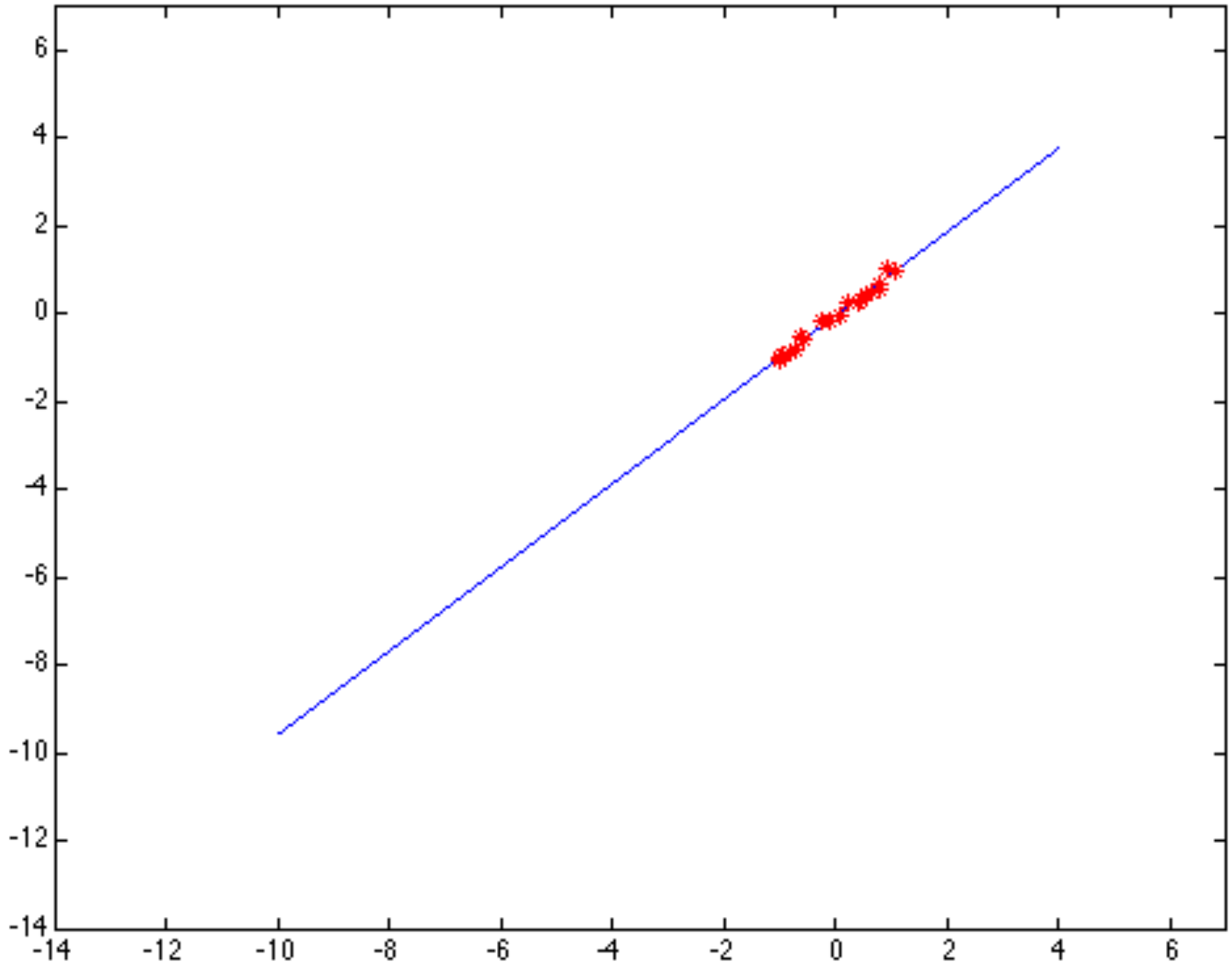
Outliers

- **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.

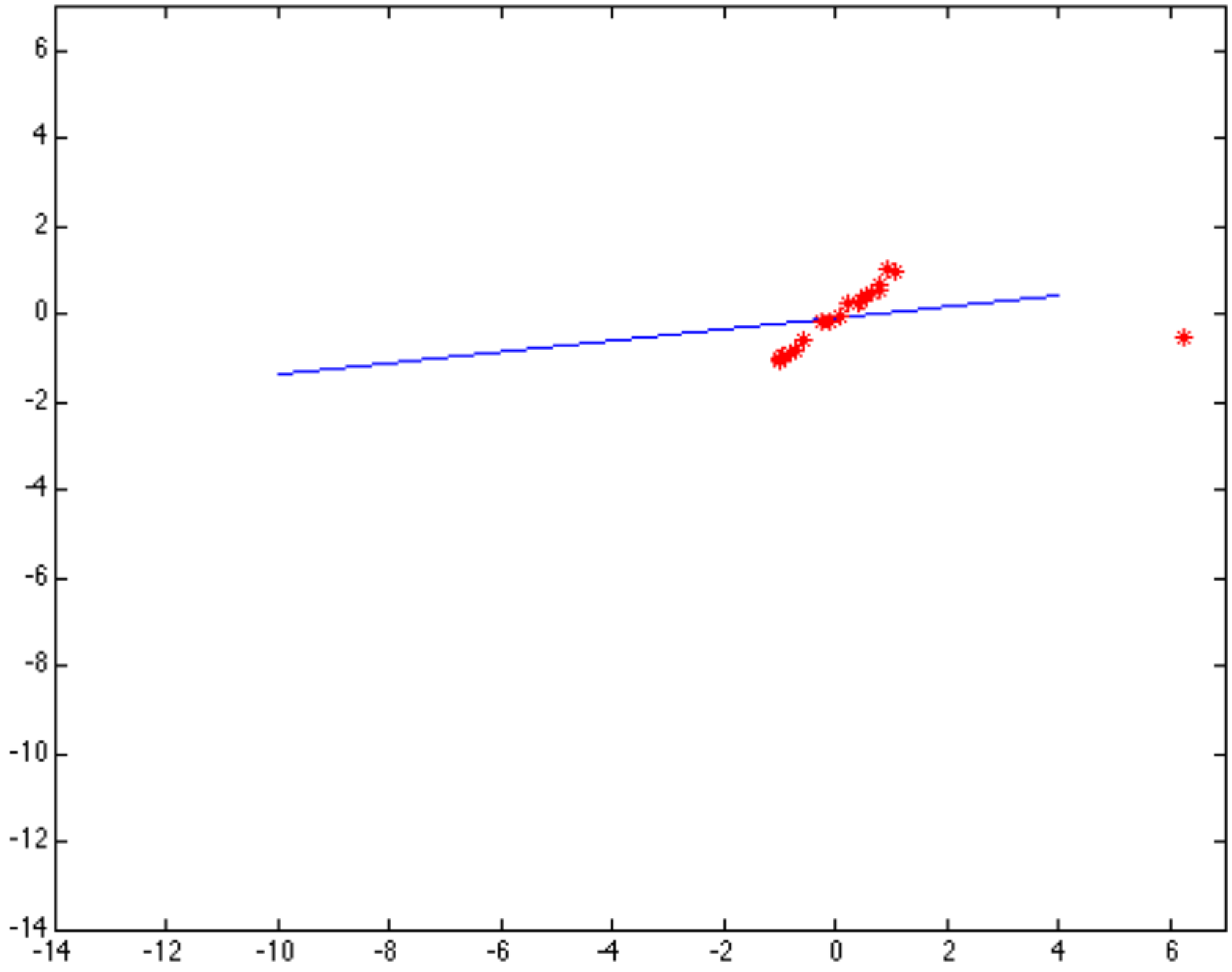


Kristen Grauman

Outliers affect least squares fit



Outliers affect least squares fit



RANSAC

- RANdom Sample Consensus

- **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 won't have much support from rest of the points.

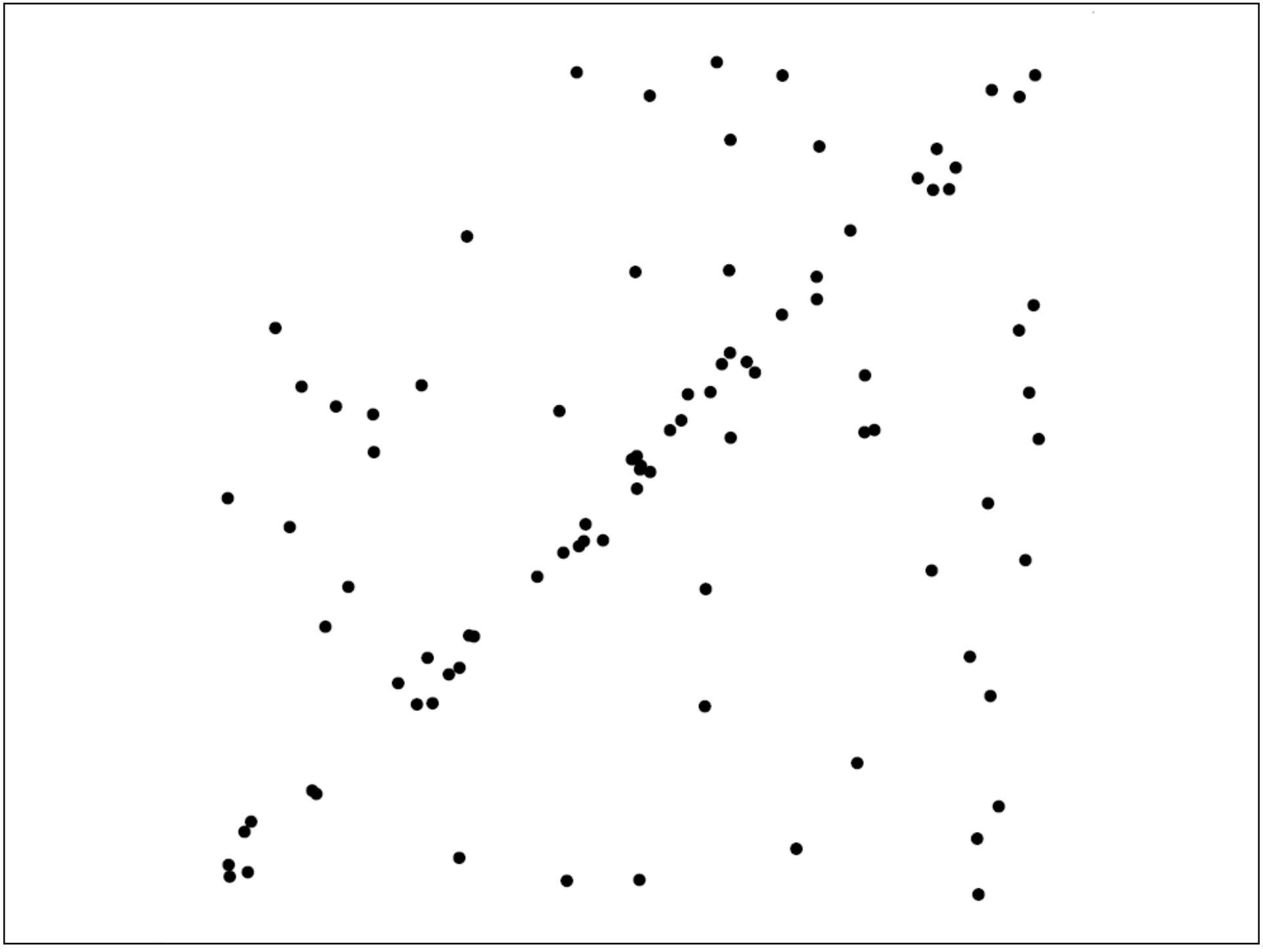
RANSAC: General form

- <u>RANSAC loop</u>:

1. Randomly select a ***seed group*** of points 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

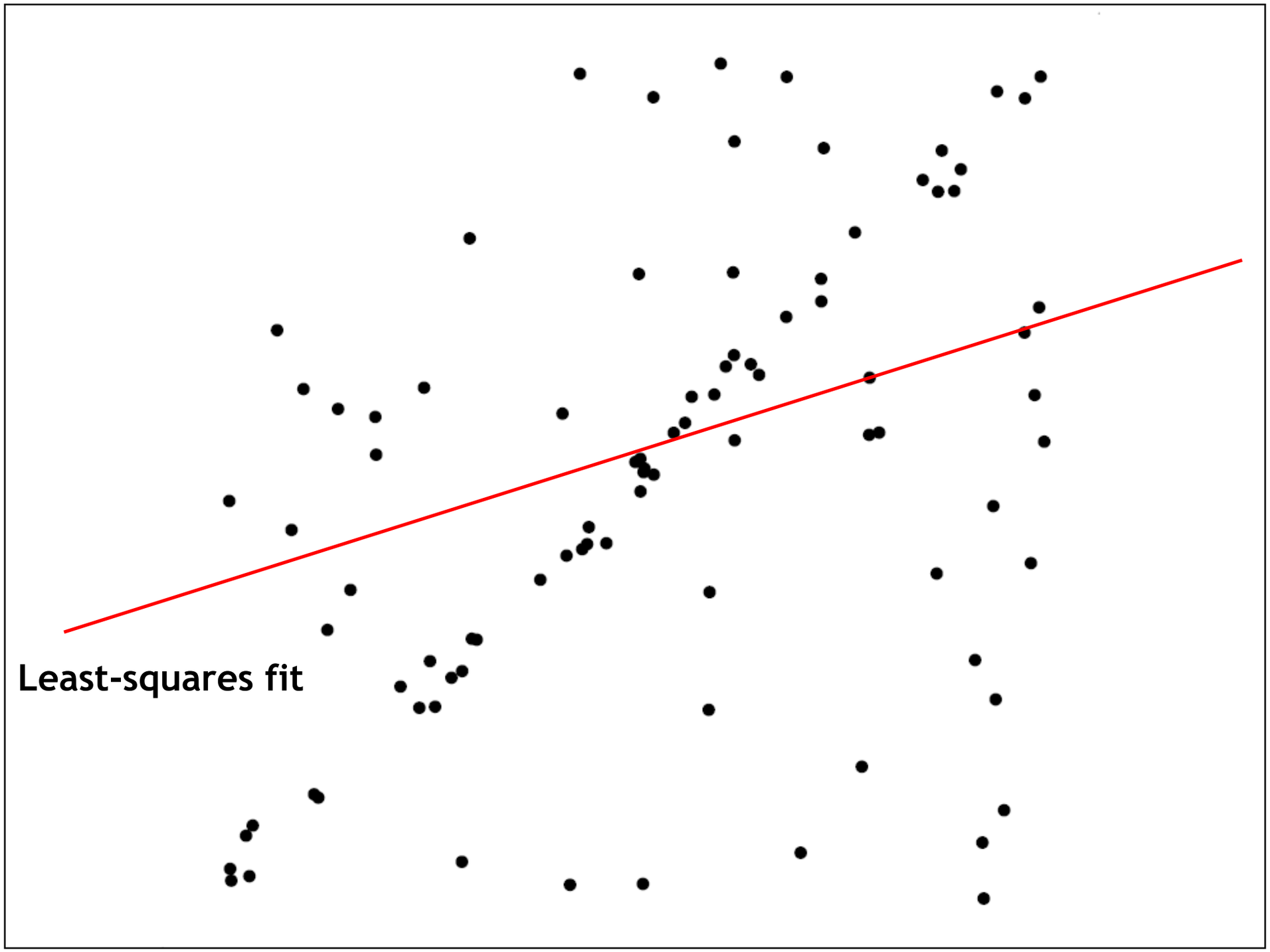
- Keep the transformation with the largest number of inliers

RANSAC for line fitting example



Source: R. Raguram

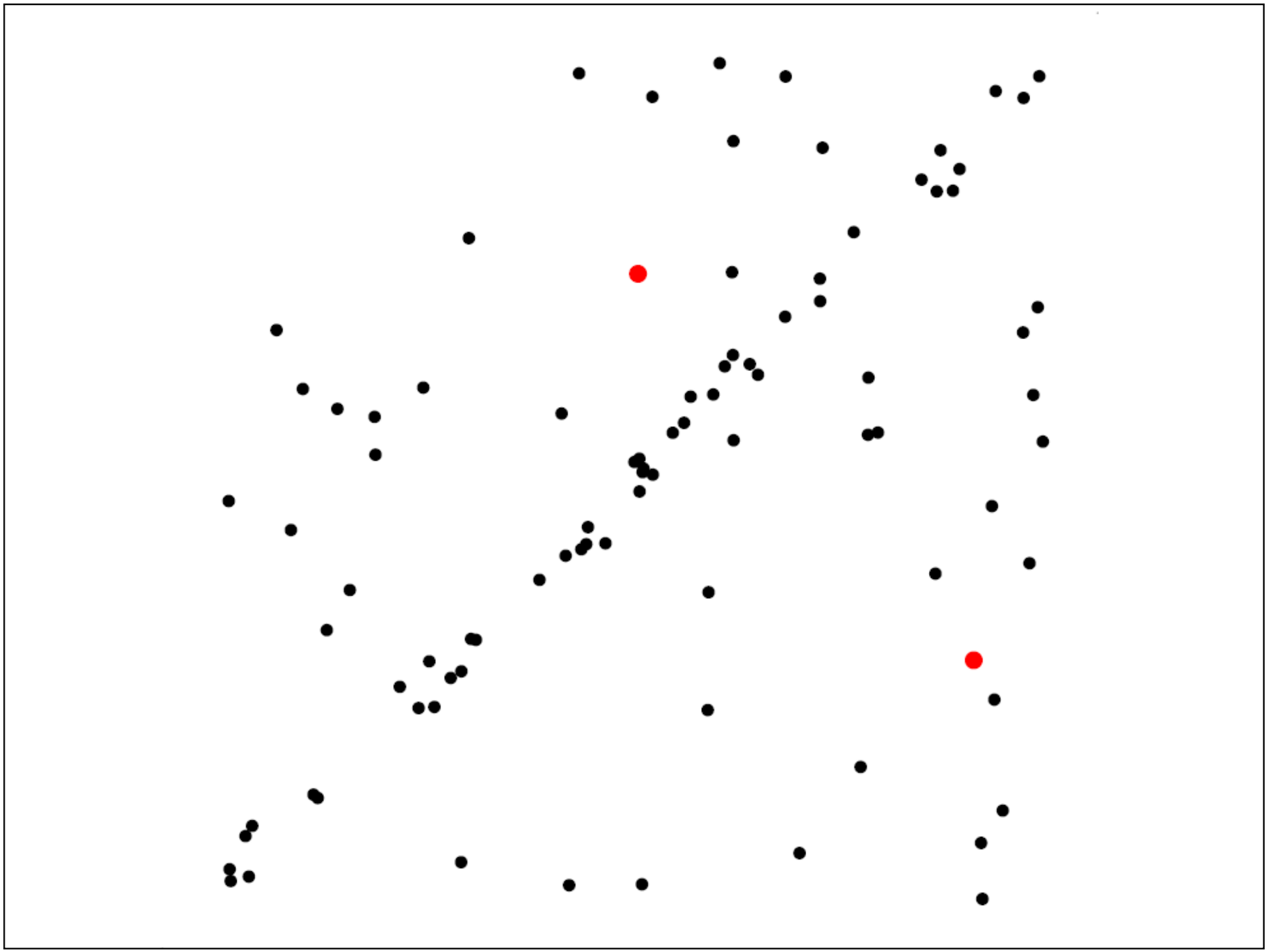
RANSAC for line fitting example



Source: R. Raguram

Lana Lazebnik

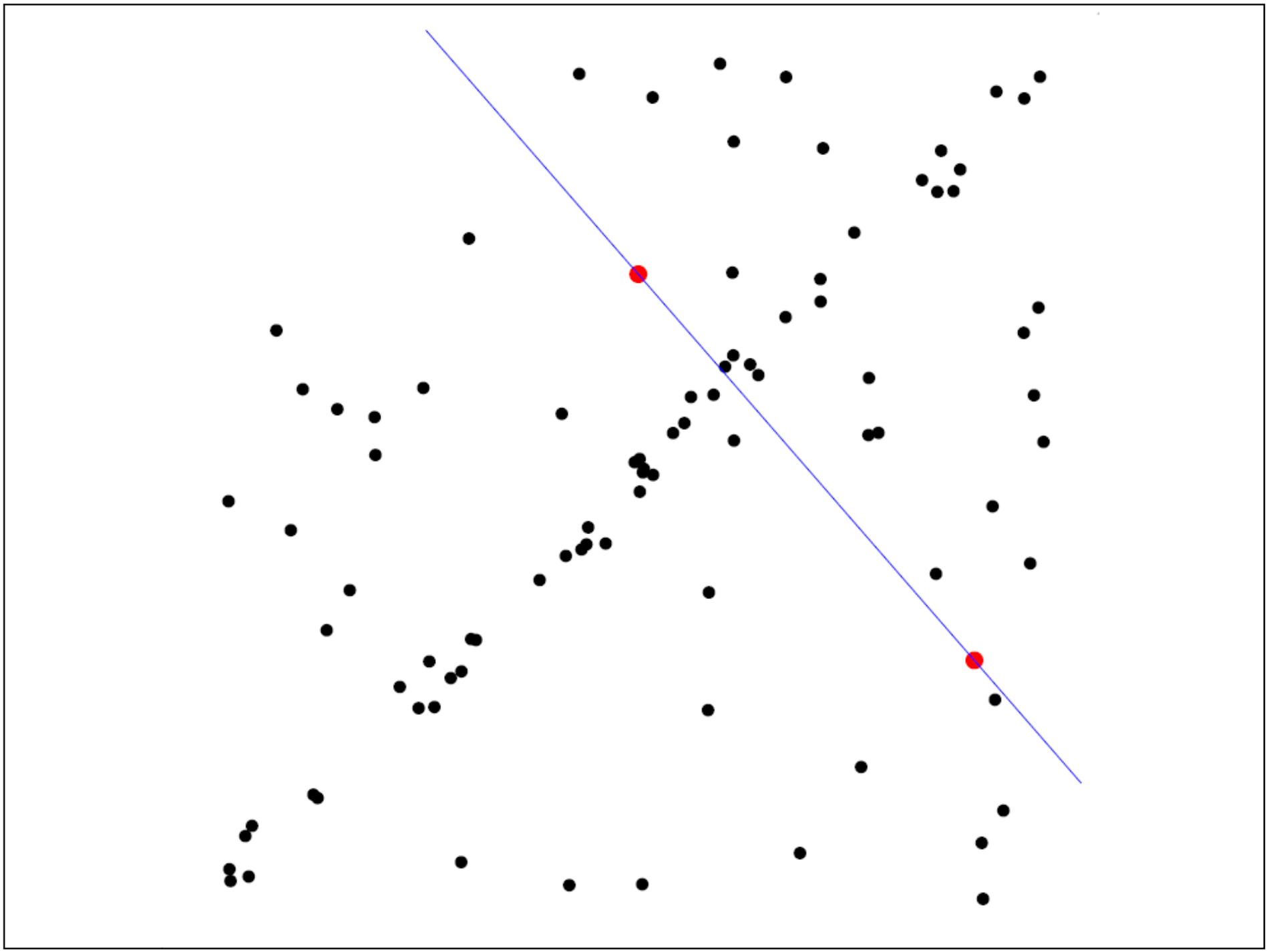
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

RANSAC for line fitting example

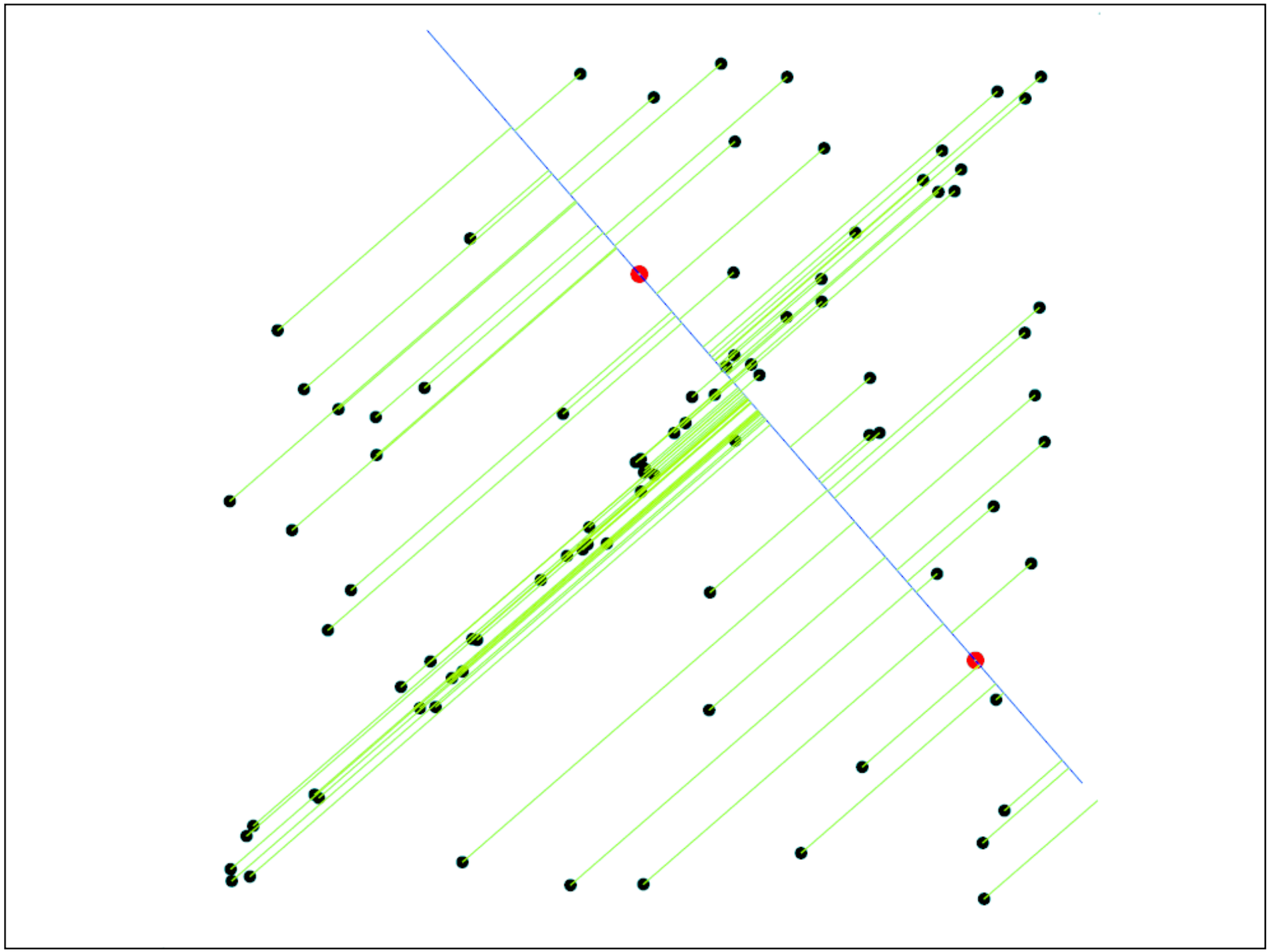


1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

Lana Lazebnik

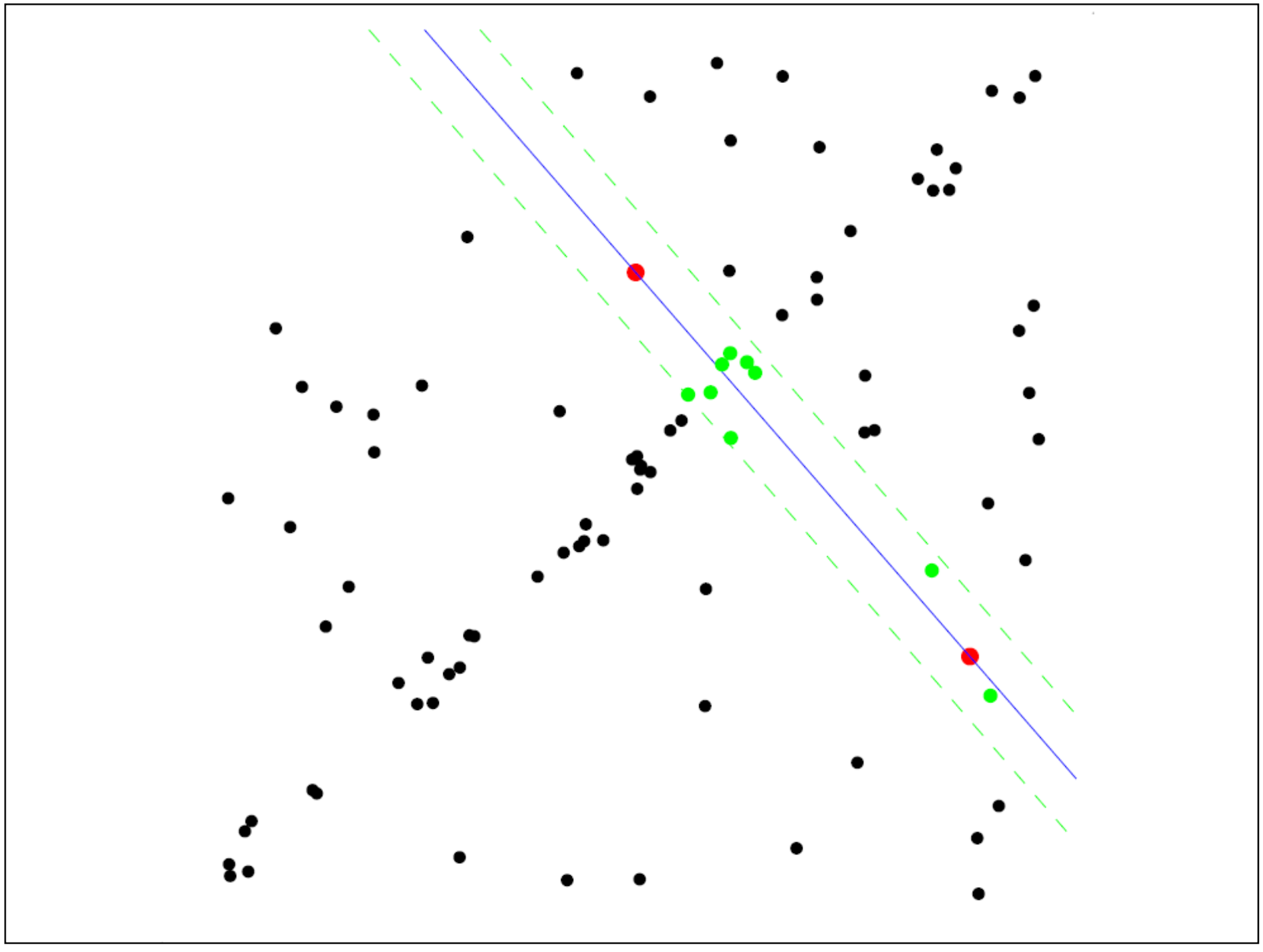
RANSAC for line fitting example



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

Source: R. Raguram

RANSAC for line fitting example

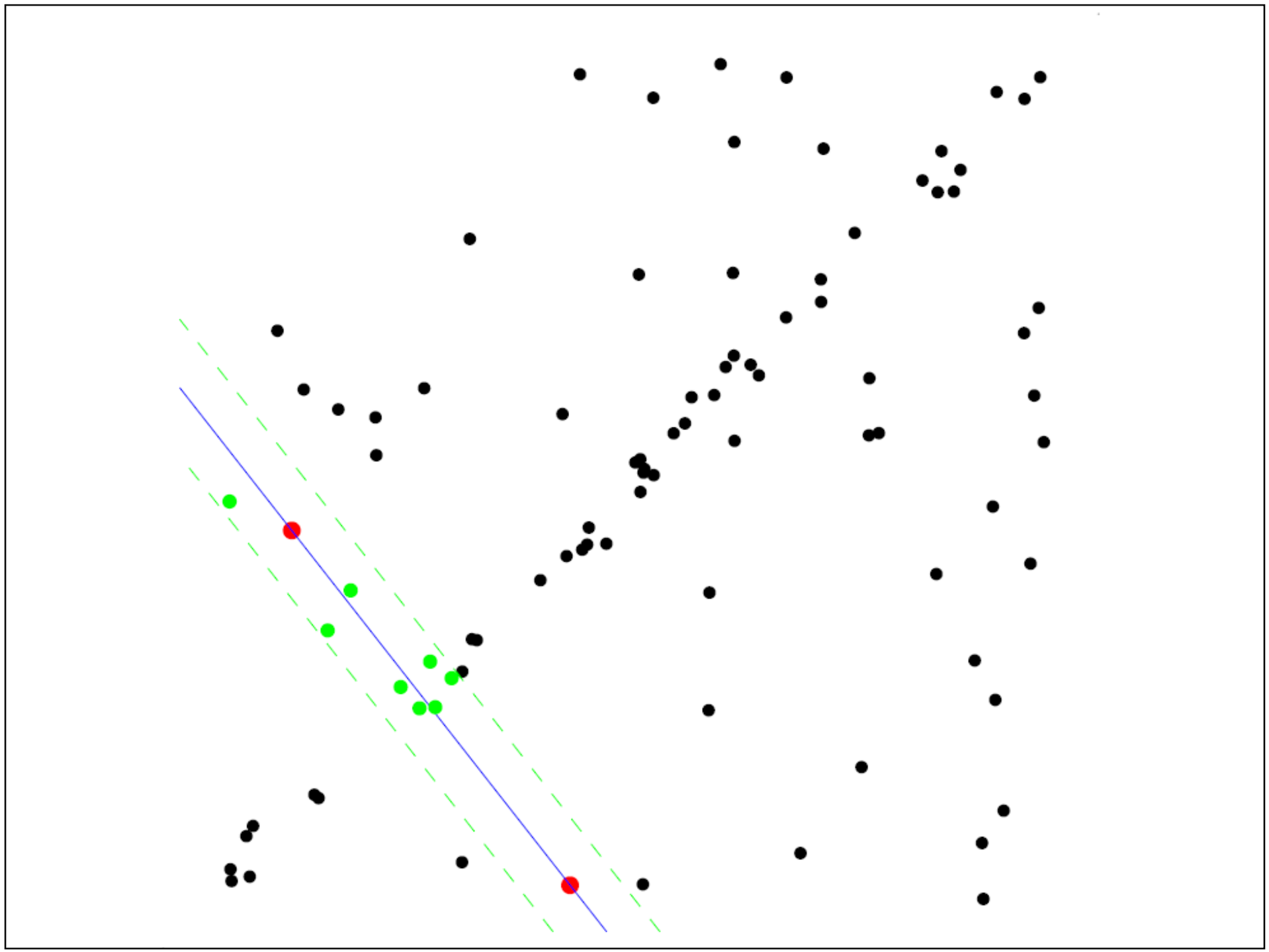


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

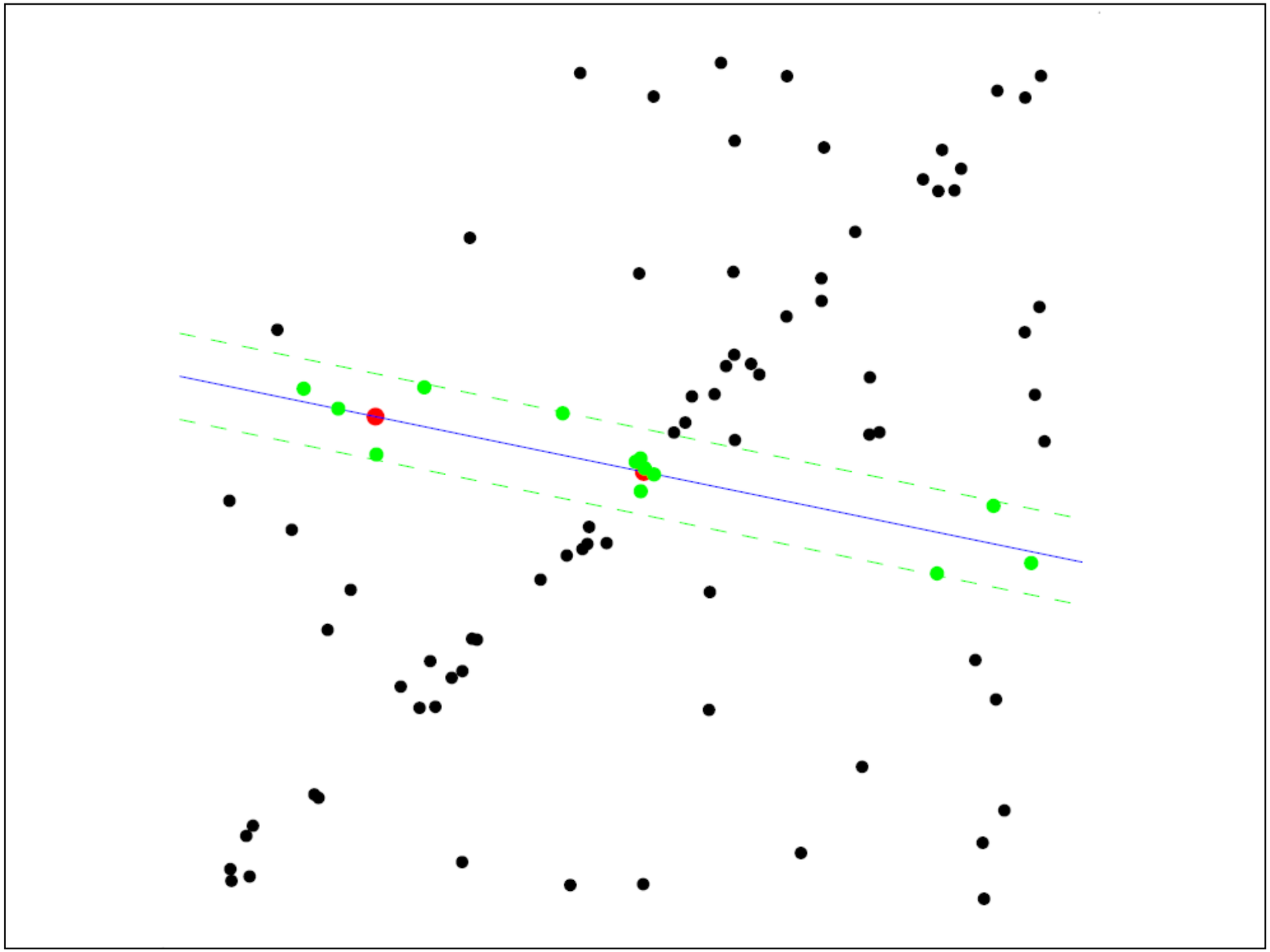


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Lana Lazebnik

RANSAC for line fitting example

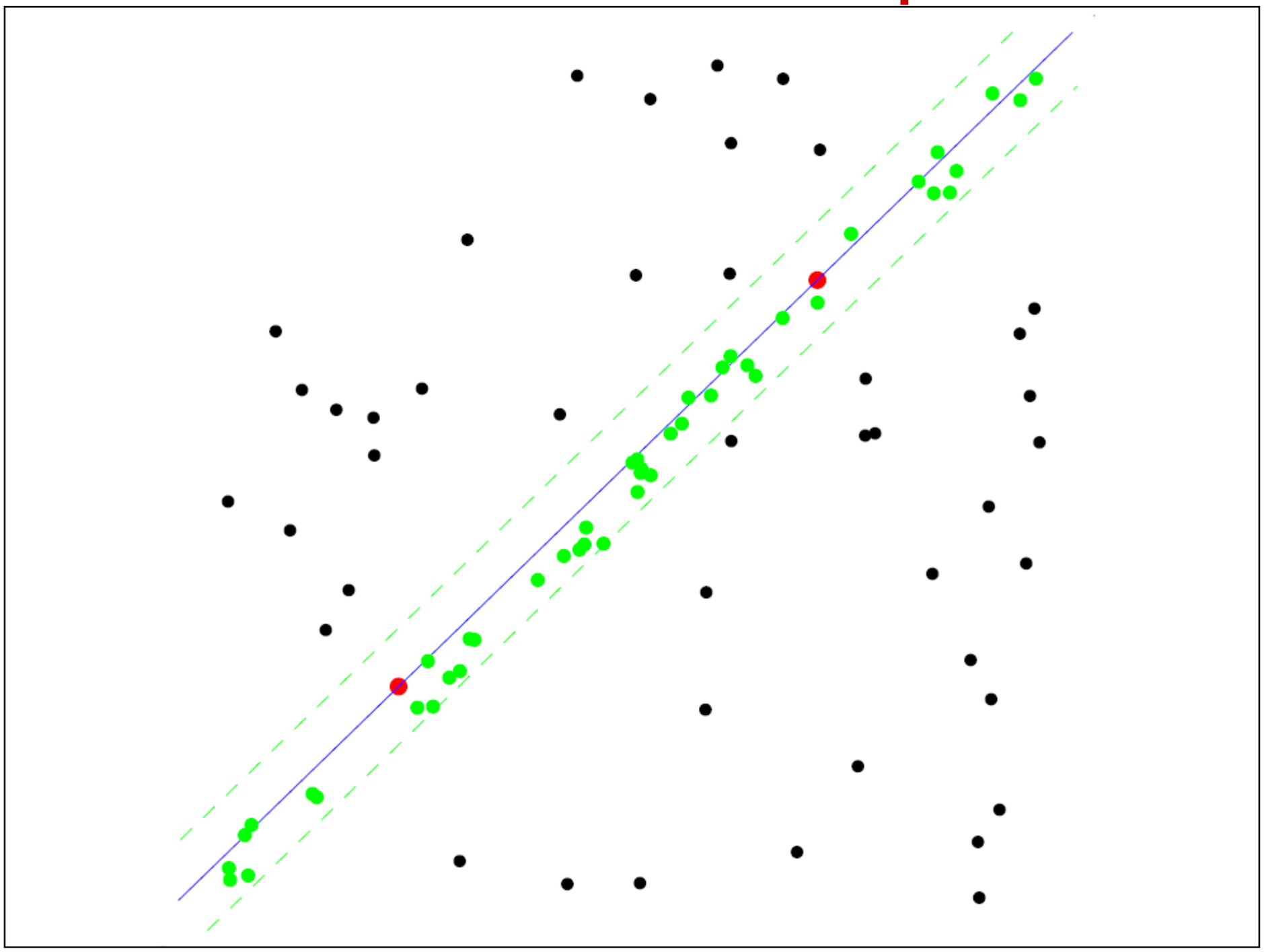


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

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

Repeat ***N*** times:

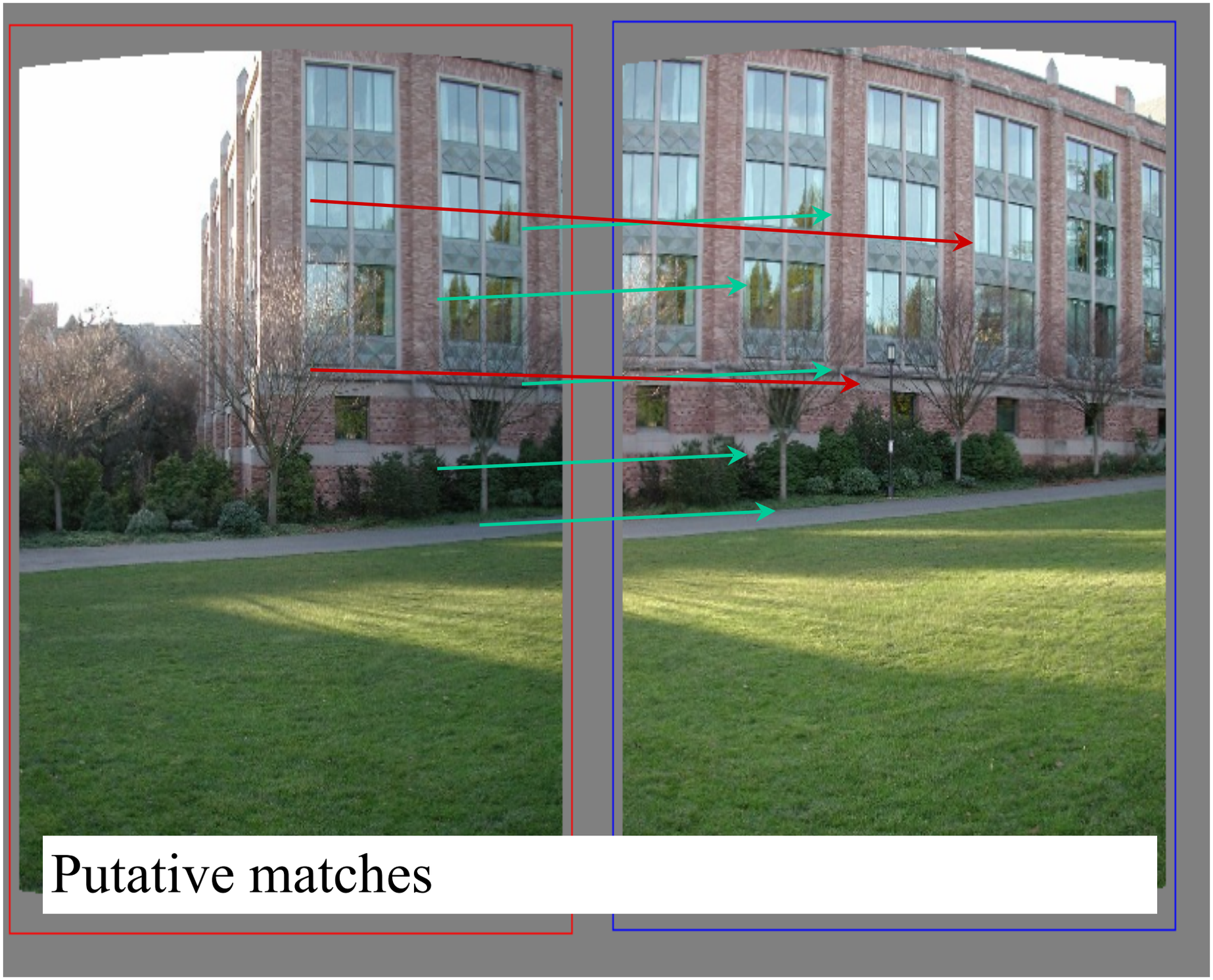
- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

That is an example fitting a line…

What about fitting a transformation (translation)?

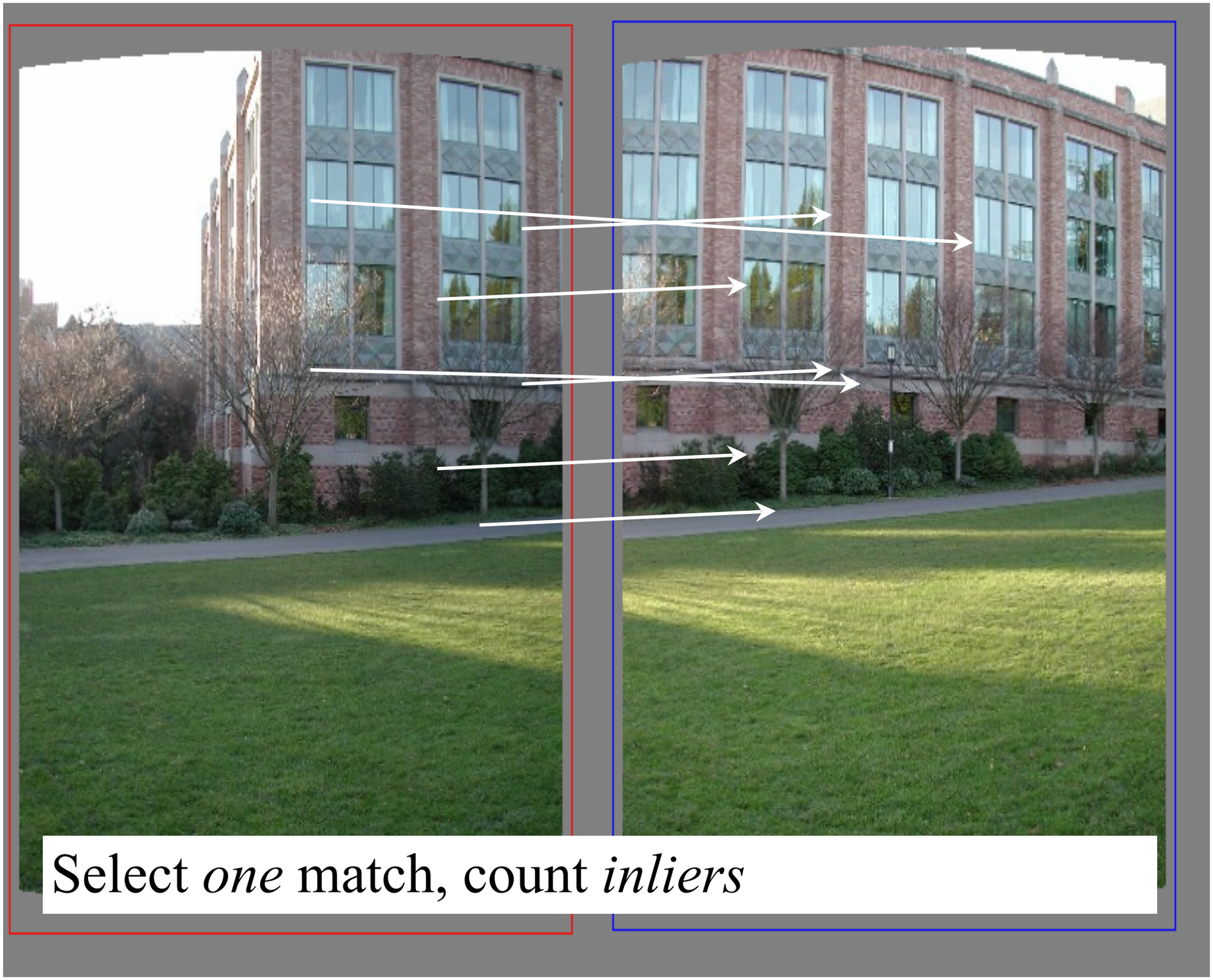
translation

RANSAC example: Translation

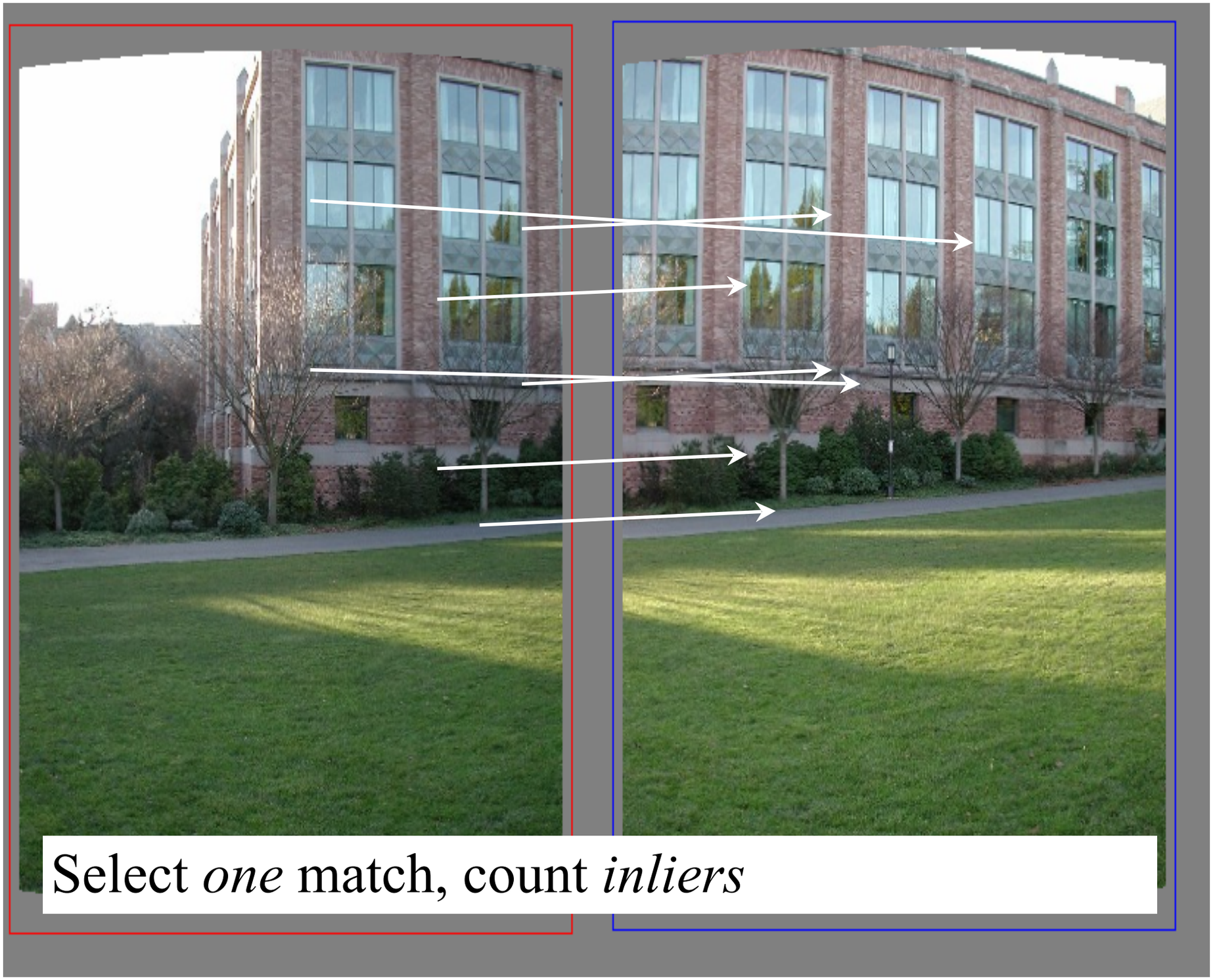


Source: Rick Szeliski

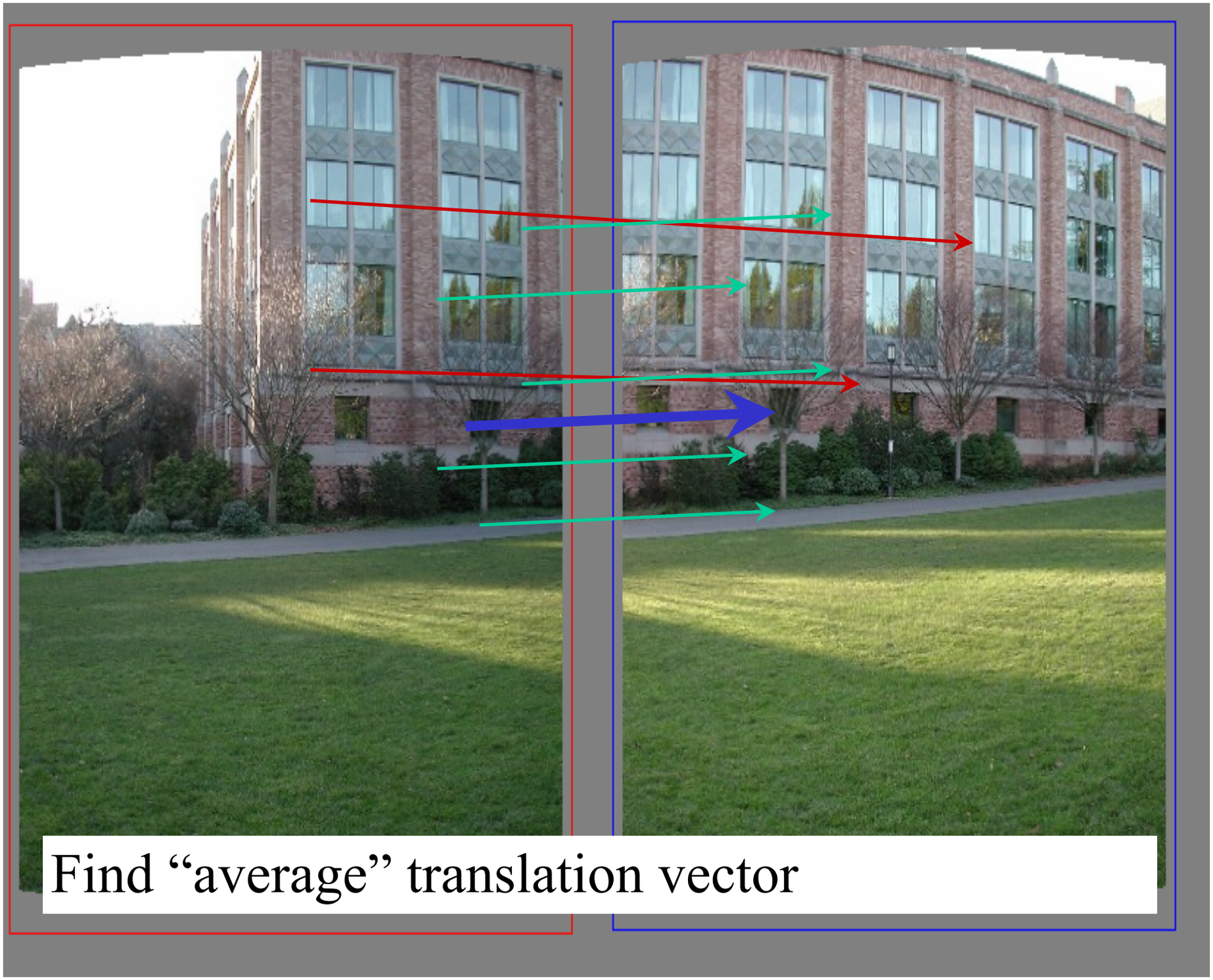
RANSAC example: Translation



RANSAC example: Translation

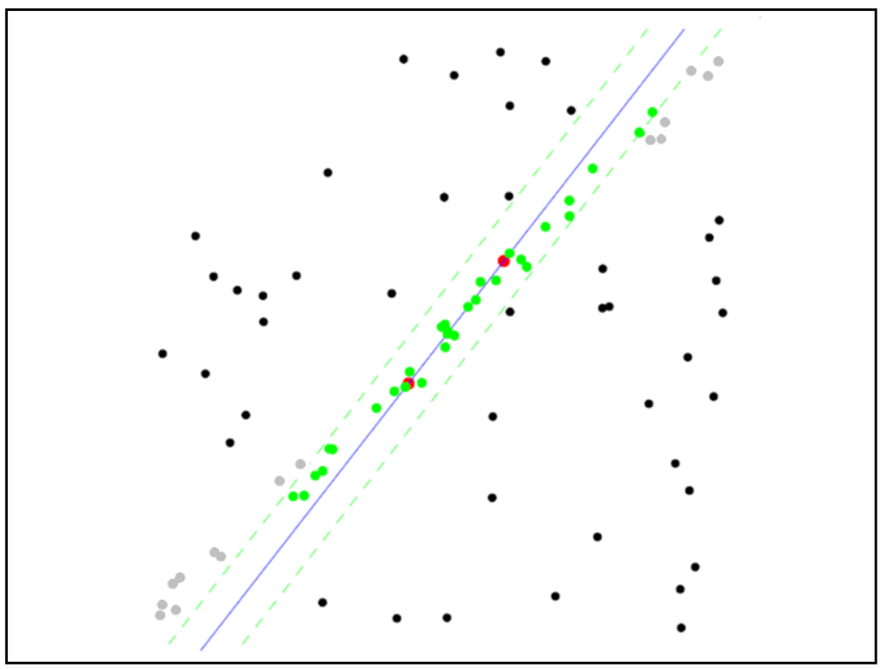


RANSAC example: Translation



RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of 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

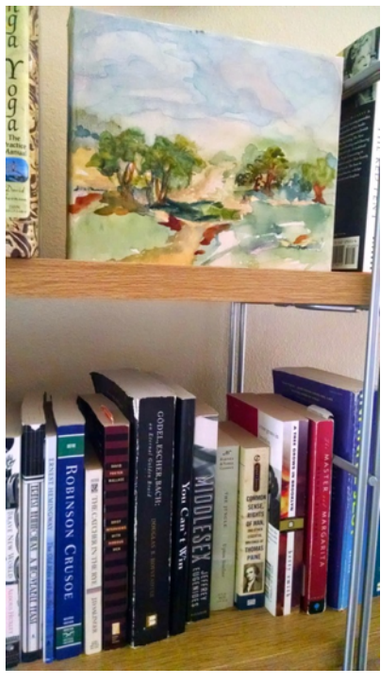


Lana Lazebnik

Goal: combine images of the same scene

+

+



−

=

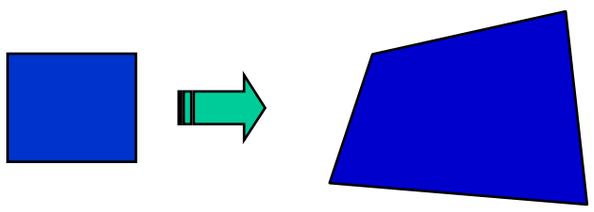
Projective Transformations

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Projective transformations:

- Affine transformations, and
- Projective warps

Parallel lines do not necessarily remain parallel



Mosaics

image from S. Seitz

Obtain a wider angle view by combining multiple images.

How to stitch together a panorama (a.k.a. mosaic)?

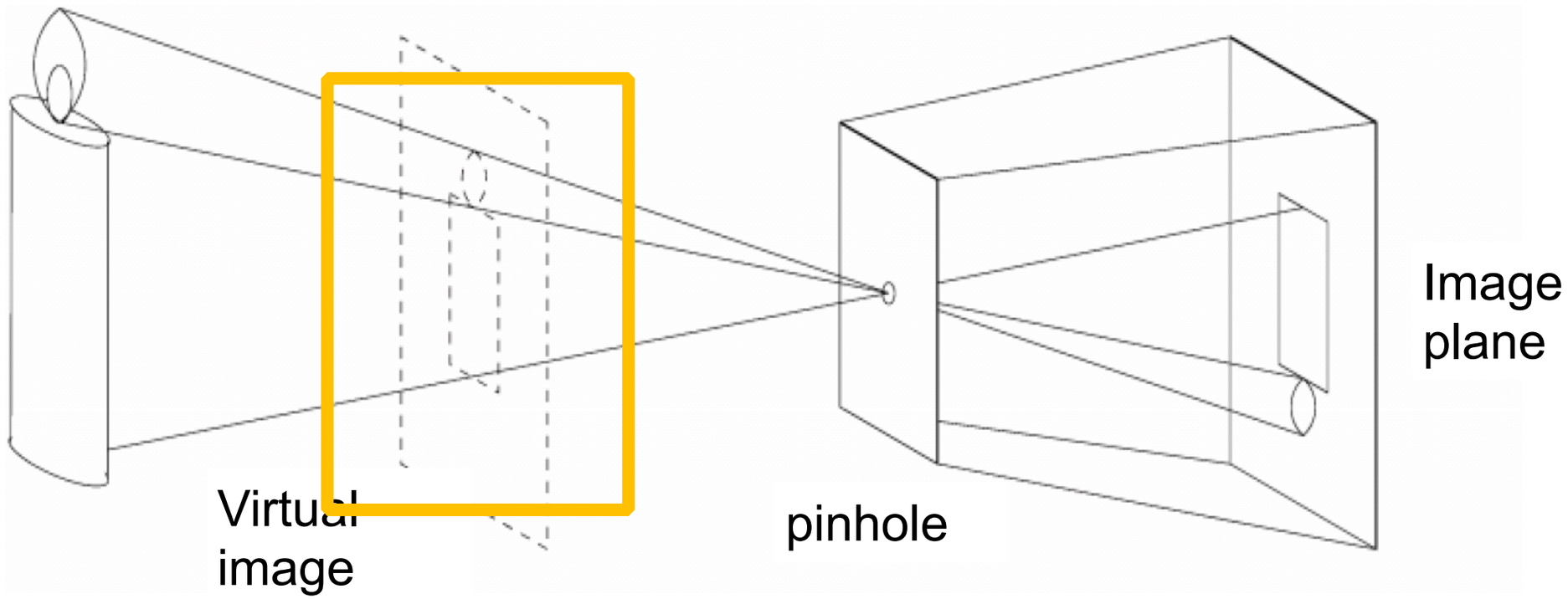
- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)

- …but **wait**, why should this work at all?
  - What about the 3D geometry of the scene?
  - Why aren't we using it?

Source: Steve Seitz

Pinhole camera

- Pinhole camera is a simple model to approximate imaging process, perspective **projection**.



If we treat pinhole as a point, only one ray from any given point can enter the camera.

Fig from Forsyth and Ponce

Mosaics

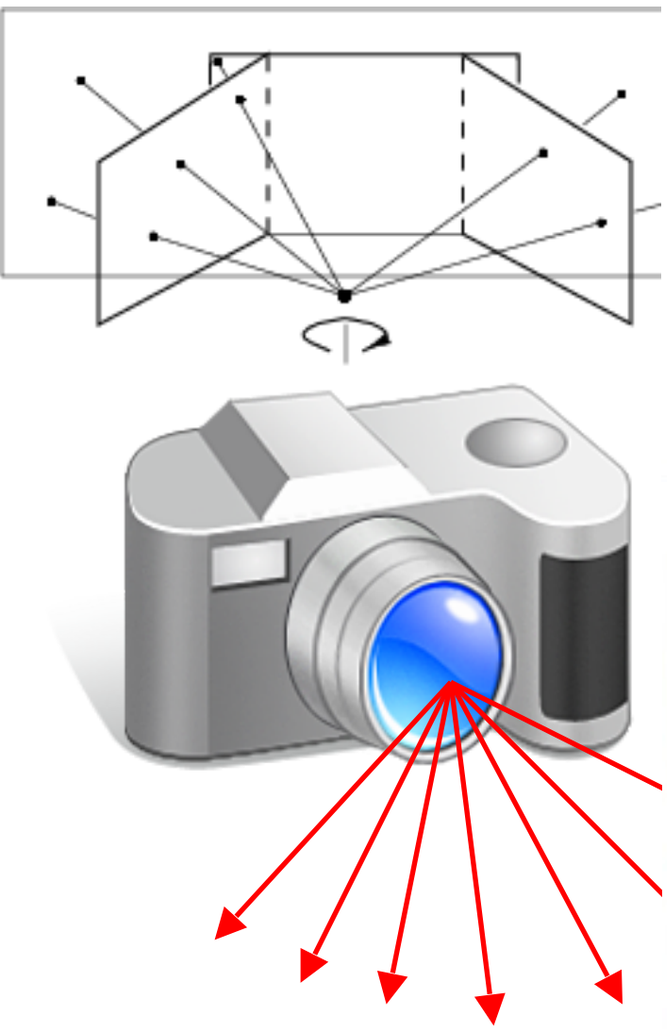
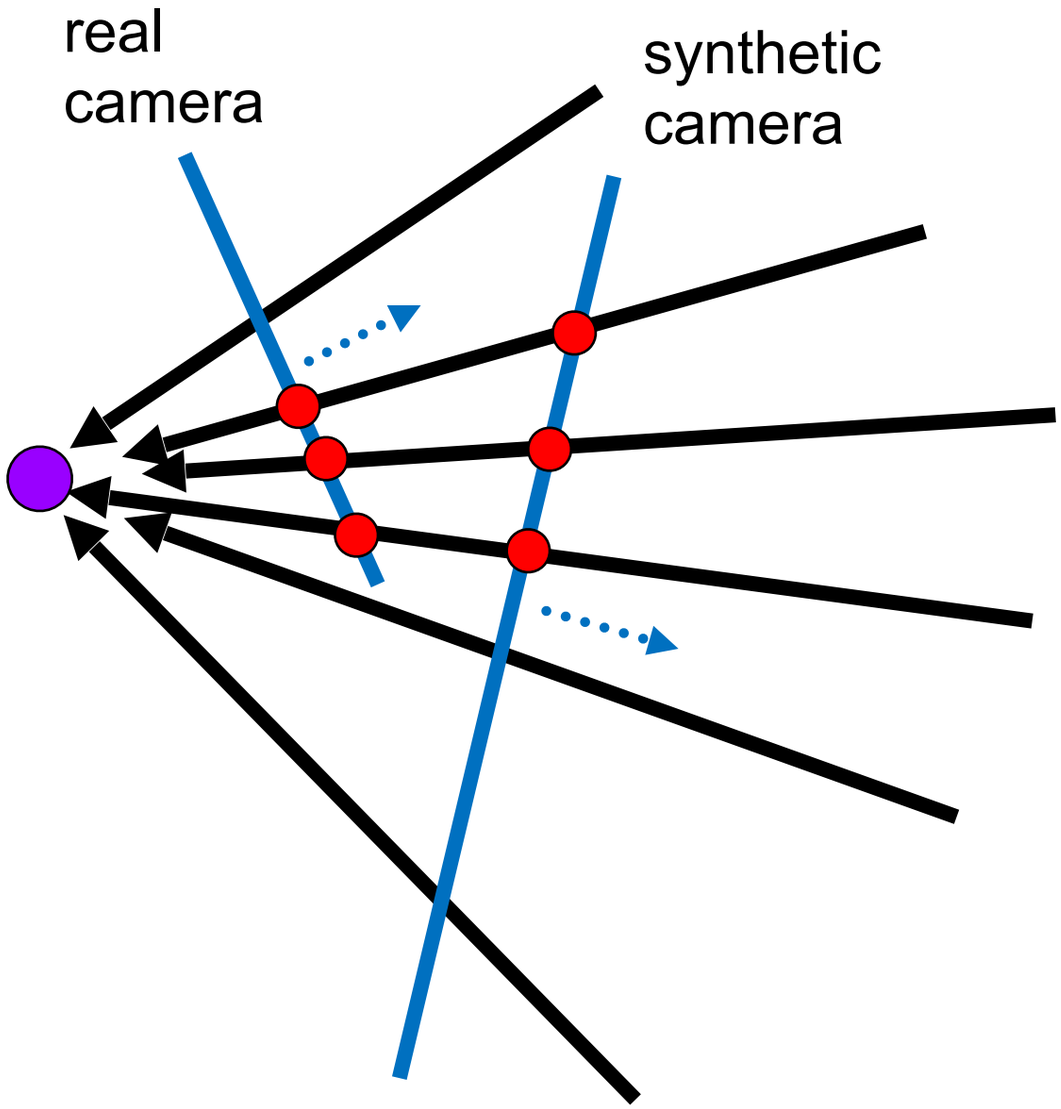


image from S. Seitz

Obtain a wider angle view by combining multiple images.

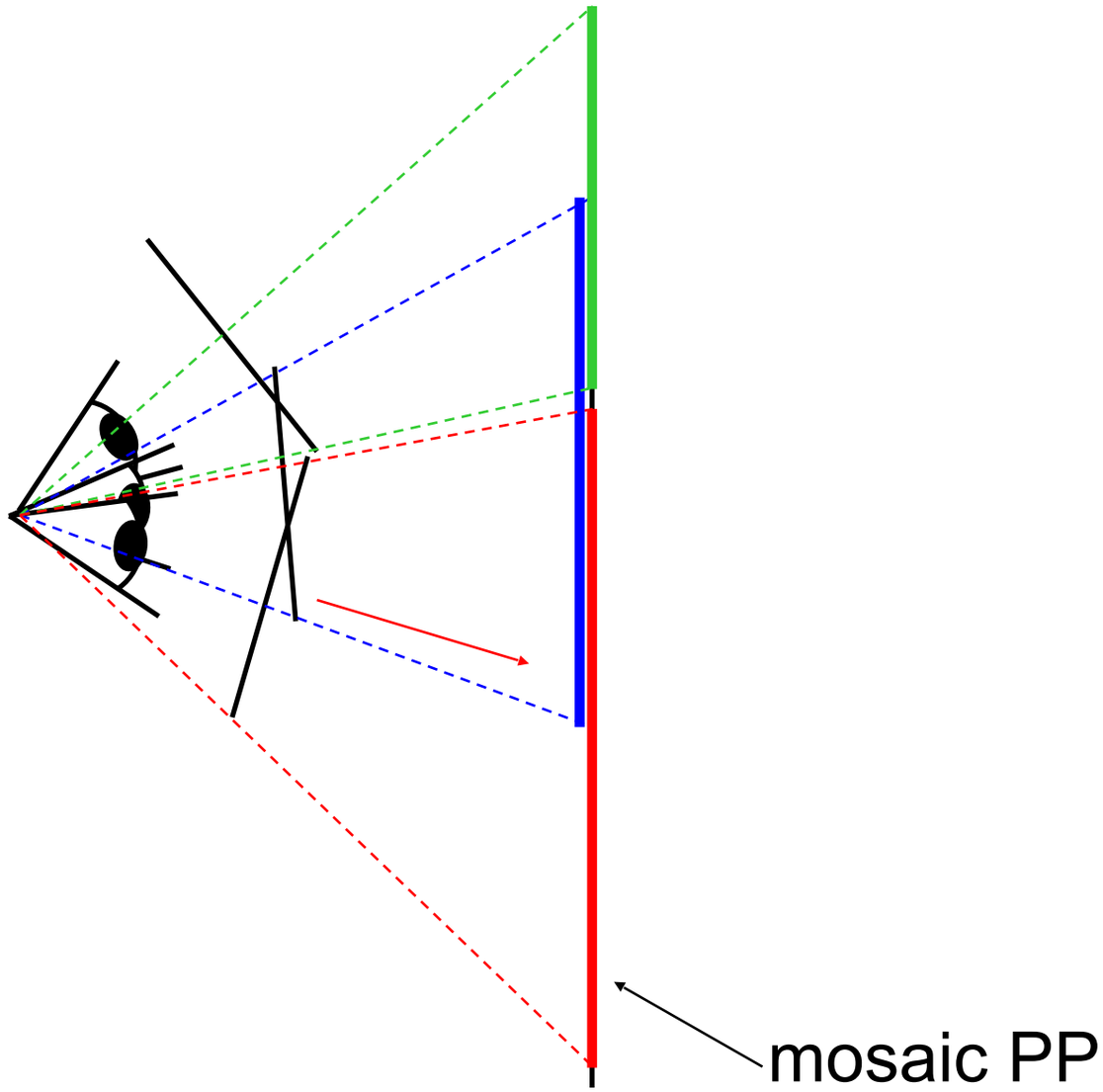
Mosaics: generating synthetic views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

Source: Alyosha Efros

Image reprojection



The mosaic has a natural interpretation in 3D

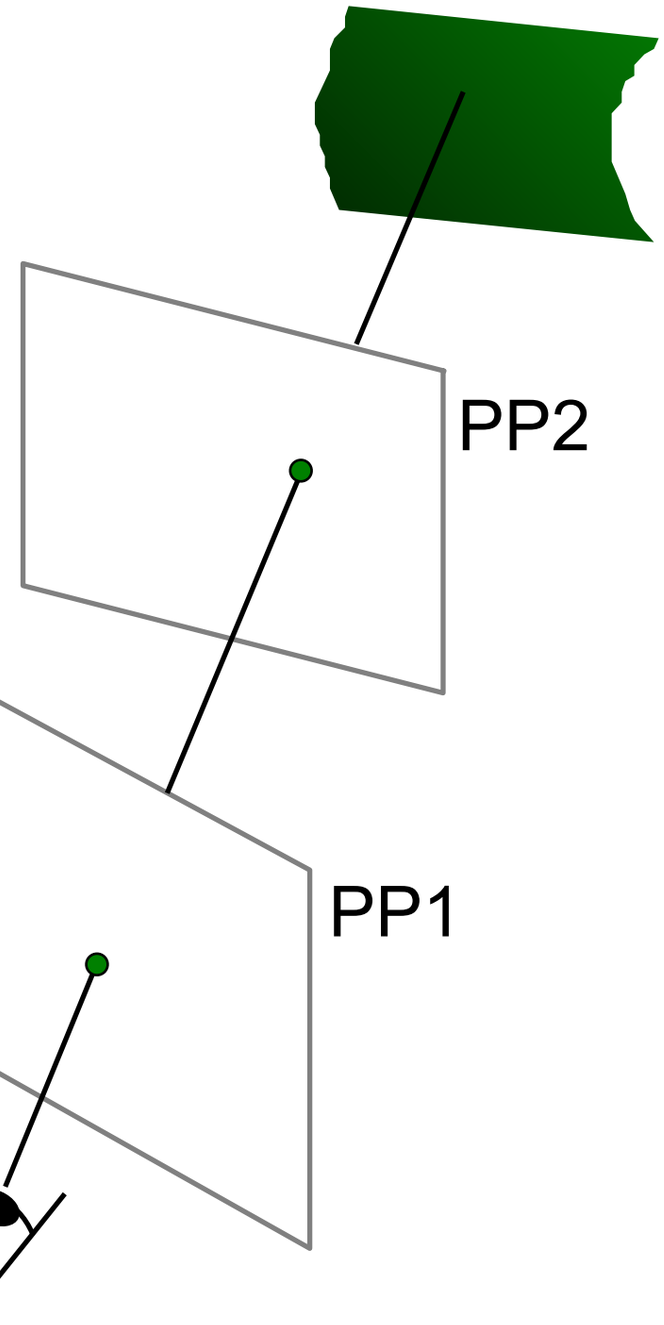
- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Source: Steve Seitz

Image reprojection

Basic question

- How to relate two images from the same camera center?
  - how to map a pixel from PP1 to PP2



Answer

- Cast a ray through each pixel in PP1
- Draw the pixel where that ray intersects PP2

Observation:
Rather than thinking of this as a 3D reprojection, think of it as a 2D **image warp** from one image to another.

Source: Alyosha Efros

Image reprojection: Homography

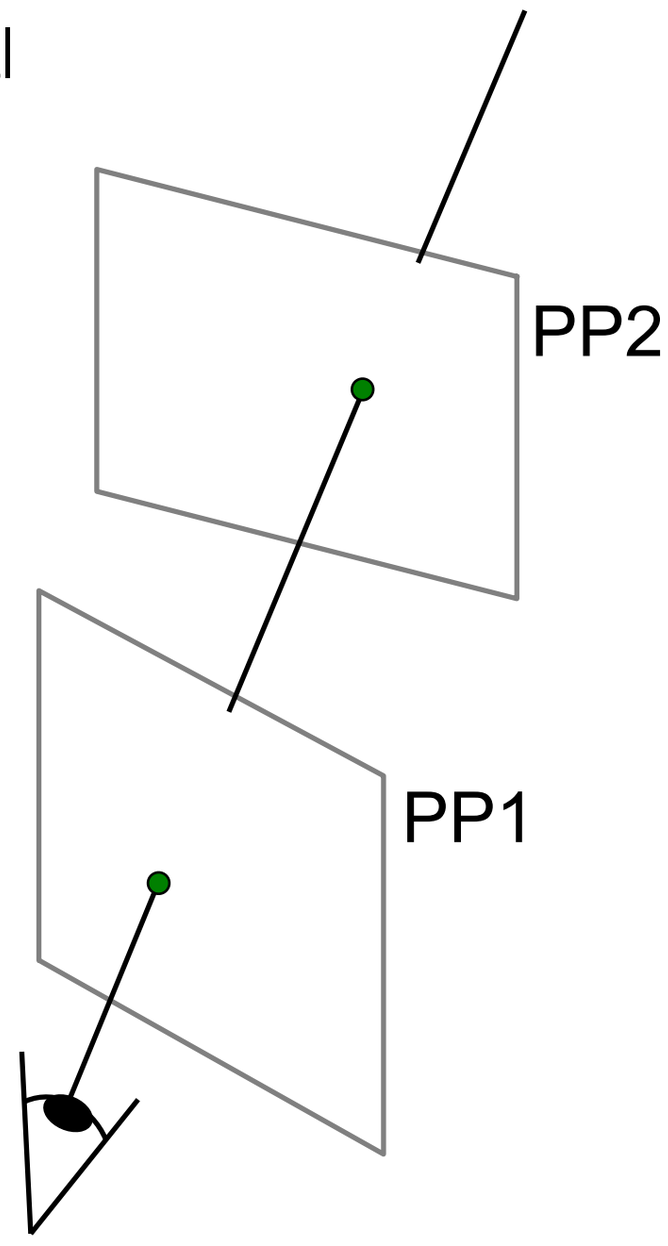
A projective transform is a mapping between any two PPs with the same center of projection

- rectangle should map to arbitrary quadrilateral
- parallel lines aren't
- but must preserve straight lines

called **Homography**

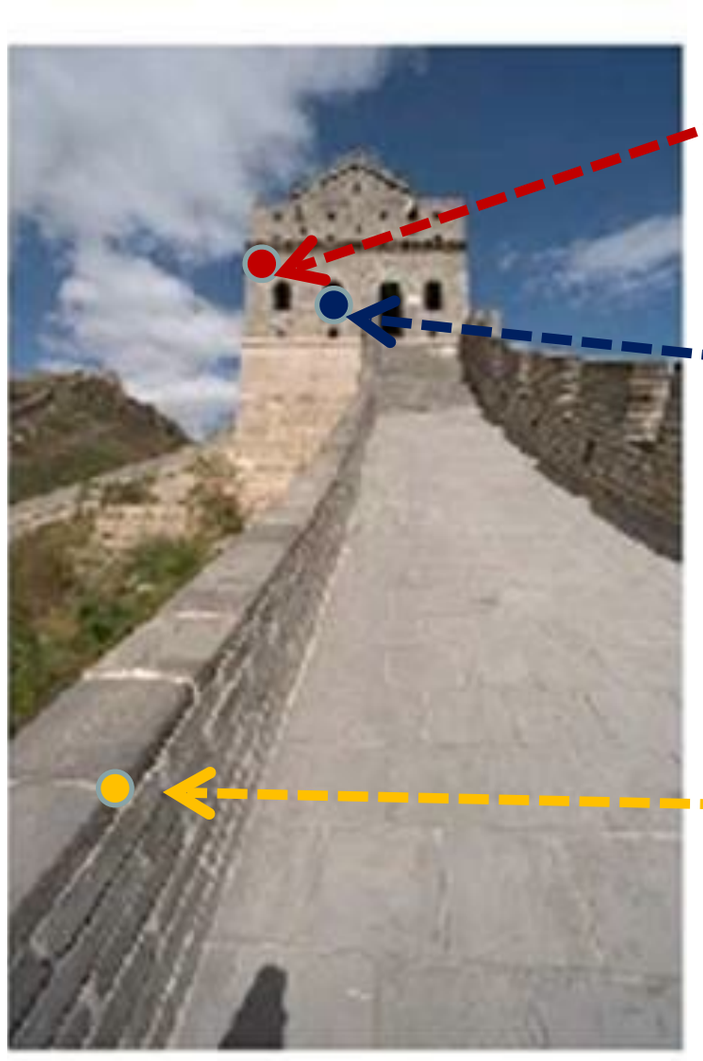
$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\mathbf{p'}$ $\mathbf{H}$ $\mathbf{p}$



Source: Alyosha Efros

Homography



(x_1, y_1)

(x_2, y_2)

$\vdots$

(x_n, y_n)

(x'_1, y'_1)

(x'_2, y'_2)

$\vdots$

(x'_n, y'_n)

To **compute** the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of **H** are the unknowns…

Solving for homographies

p’ = Hp

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Can set scale factor *i*=1. So, there are 8 unknowns.

Set up a system of linear equations:

Ah = b

where vector of unknowns h = [a,b,c,d,e,f,g,h]^T

Need at least 8 eqs, but the more the better…

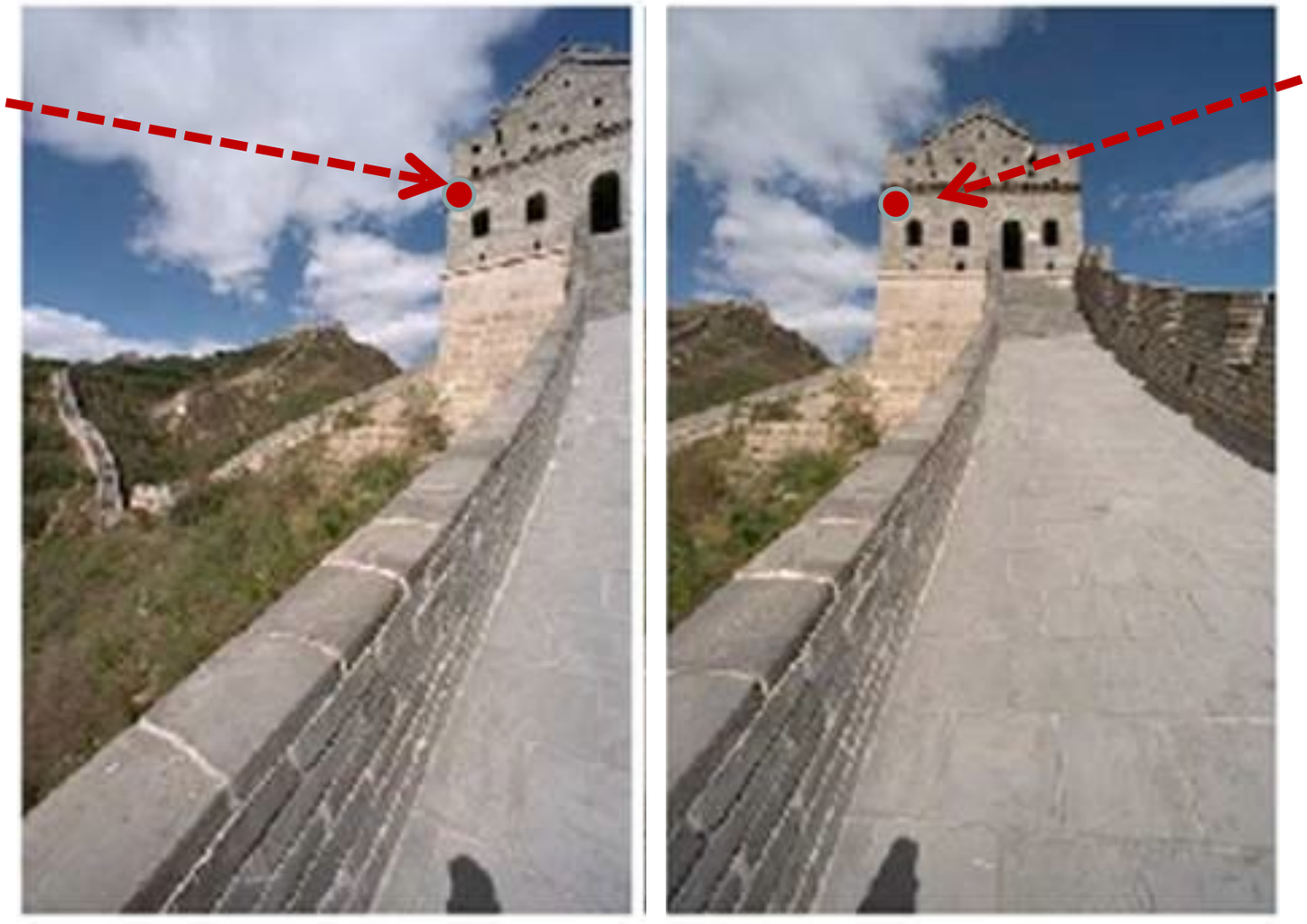
Solve for h. If overconstrained, solve using least-squares:

$$\min \|Ah - b\|^2$$

Implemented by cv2.findHomography()

Homography

(x, y)



$\left({}^{wx'}/_{w}, {}^{wy'}/_{w} \right) = (x', y')$

To **apply** a given homography **H**

- Compute **p’** = **Hp** (regular matrix multiply)
- Convert **p’** from homogeneous to image coordinates

$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

Project 1
Out !

Work time:
GitHub Setup + Project 0