Image alignment

370: Intro to Computer Vision

Subhransu Maji

March 27 & April 1

College of
INFORMATION AND
COMPUTER SCIENCES

UMASS
AMHERST

A framework for alignment

Matching local features

- Local information used, can contain outliers
- But hopefully enough of these matches are good

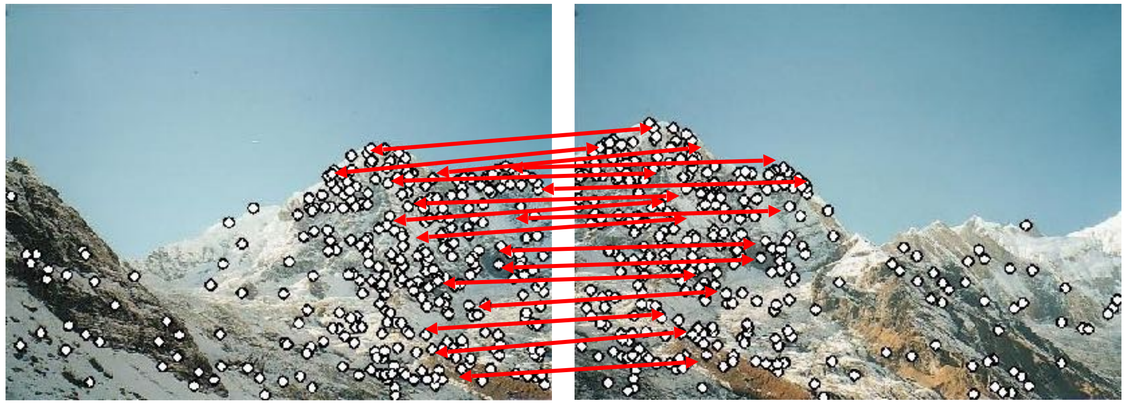
A framework for alignment

Matching local features

- Local information used, can contain outliers
- But hopefully enough of these matches are good

Consensus building

- Aggregate the good matches and find a transformation that explains these matches



Harris corner detector

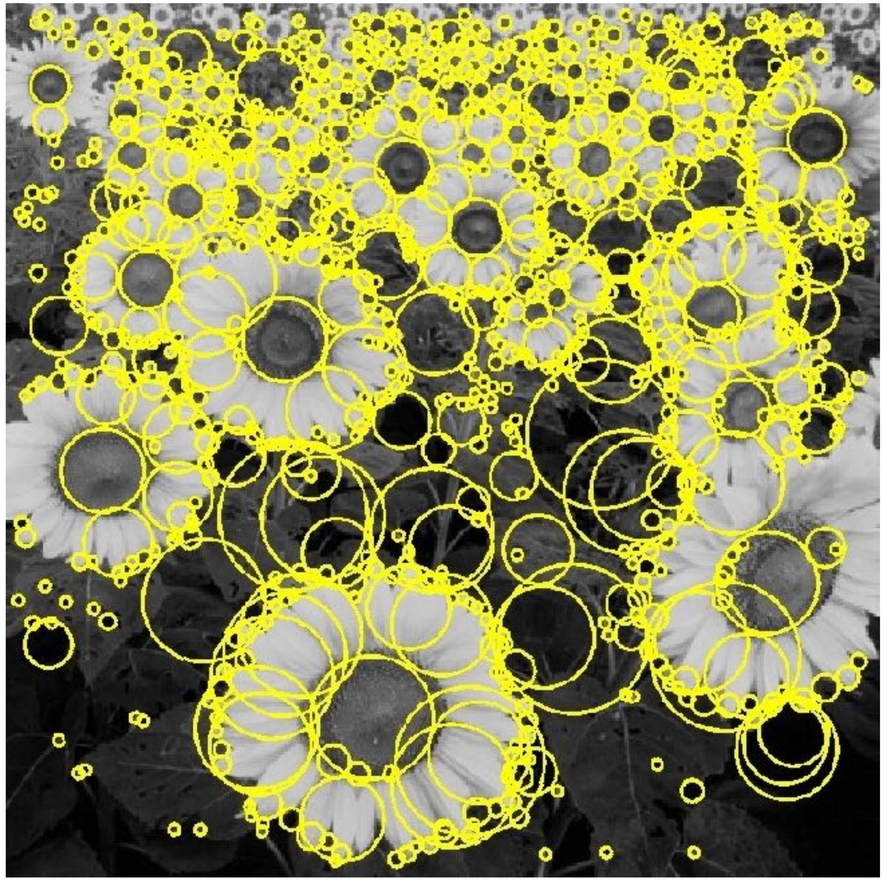
Covariance to translation of the object

- If the object moves by **t** the corners also move by **t**

Scale covariant features

Scale “covariance”

“blob” detection



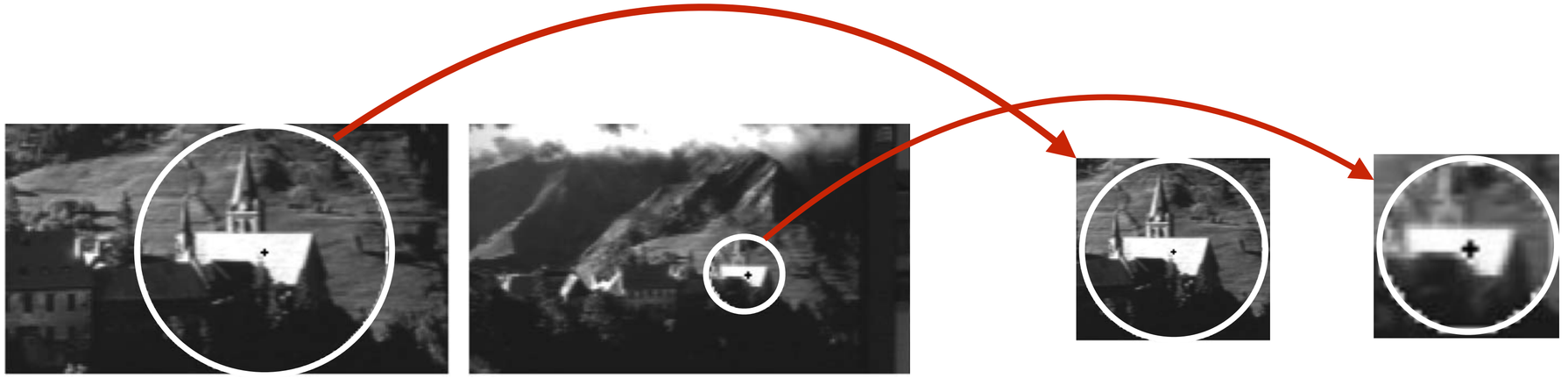
Source: L. Lazebnik

From feature detection to description

Scaled and translated versions of the same neighborhood will give rise to blobs that are related by the same transformation

What to do if we want to compare the appearance of these image regions?

- **Normalization**: transform these regions into same-size circles
- Problem: rotational ambiguity

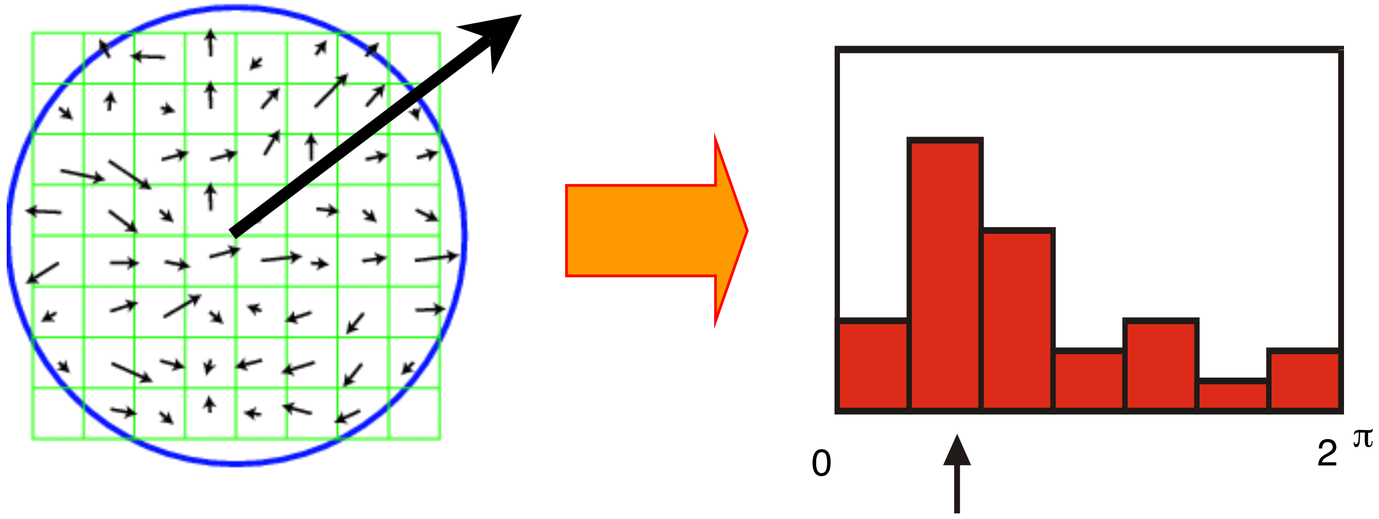


Source: L. Lazebnik

Eliminating rotation ambiguity

To assign a unique orientation to circular image windows:

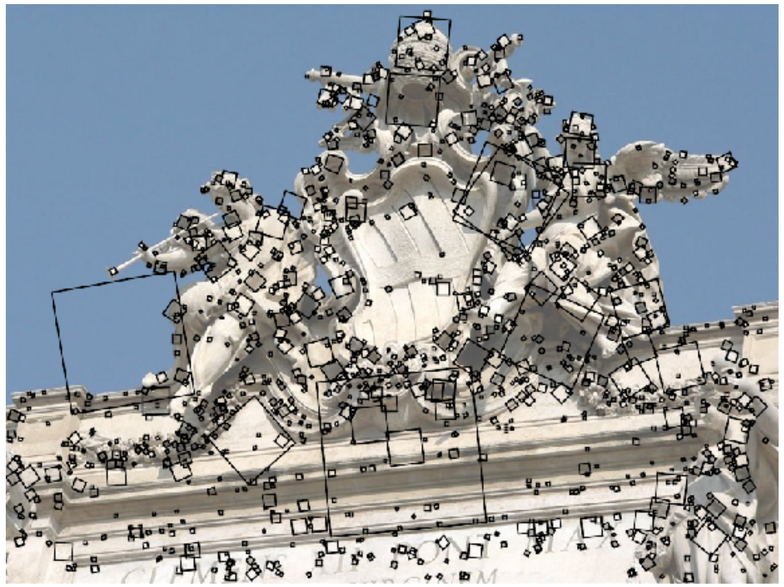
- Create histogram of local gradient directions in the patch
- Assign canonical orientation at peak of smoothed histogram



Source: L. Lazebnik

SIFT features

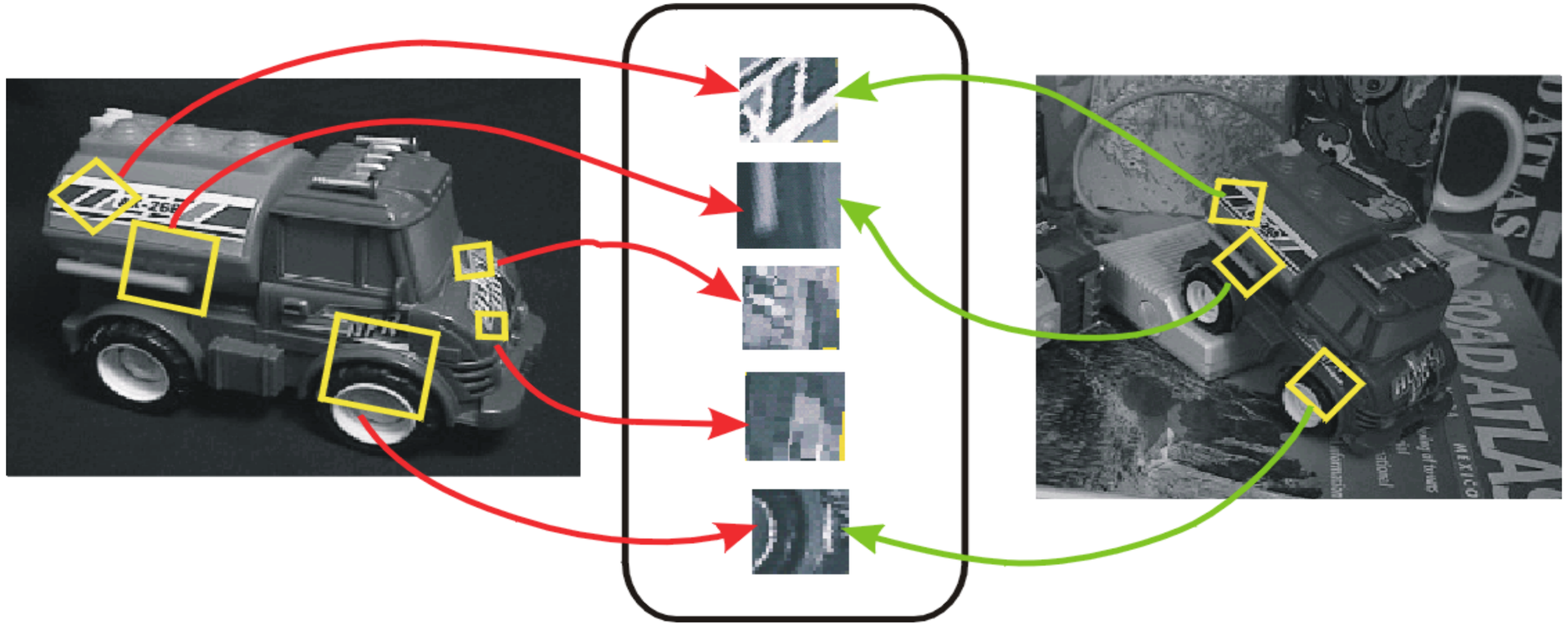
Detected features with characteristic scales and orientations:



David G. Lowe. **"Distinctive image features from scale-invariant keypoints."** *IJCV* 60 (2), pp. 91-110, 2004.

Source: L. Lazebnik

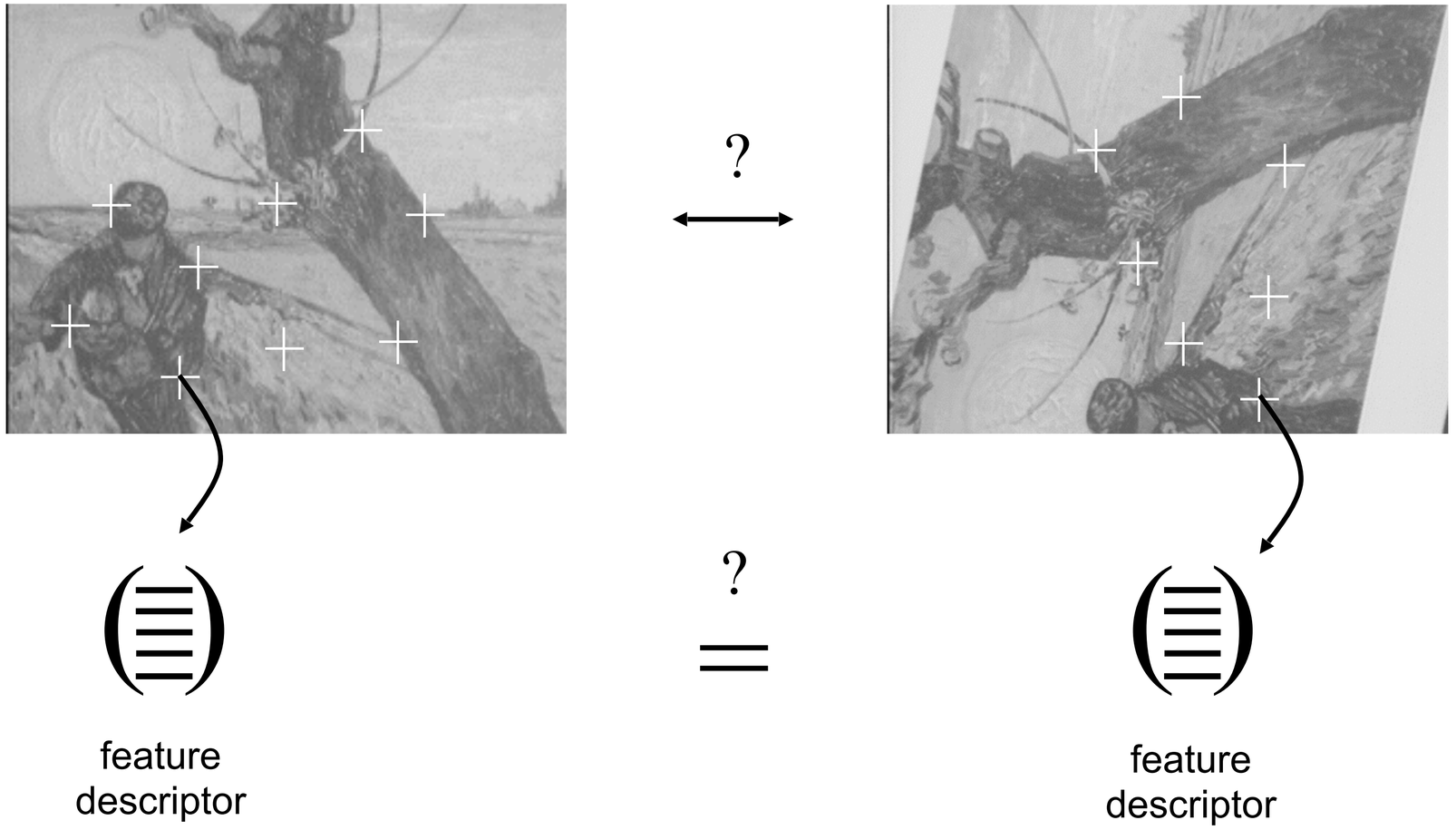
From feature detection to description



How should we *represent* patches for matching?

Generating putative correspondences

Need to find regions and compare their feature descriptors



Feature descriptors

Simplest descriptor: vector of raw intensity values

How to compare two such vectors?

- Sum of squared differences (SSD) — this is a distance measure

$$\text{SSD}(\mathbf{u}, \mathbf{v}) = \sum_i (u_i - v_i)^2$$

Not invariant to intensity change

Feature descriptors

Simplest descriptor: vector of raw intensity values

How to compare two such vectors?

- Sum of squared differences (SSD) — this is a distance measure

$$\mathrm{SSD}(\mathbf{u}, \mathbf{v}) = \sum_i (u_i - v_i)^2$$

Not invariant to intensity change

- Normalized correlation — this is a similarity measure

$$\rho(\mathbf{u}, \mathbf{v}) = \frac{(\mathbf{u} - \overline{\mathbf{u}})}{\| \mathbf{u} - \overline{\mathbf{u}} \|} \cdot \frac{(\mathbf{v} - \overline{\mathbf{v}})}{\| \mathbf{v} - \overline{\mathbf{v}} \|} = \frac{\sum_i (u_i - \overline{\mathbf{u}})(v_i - \overline{\mathbf{v}})}{\sqrt{\left(\sum_j (u_j - \overline{\mathbf{u}})^2 \right) \left(\sum_j (v_j - \overline{\mathbf{v}})^2 \right)}}$$

Invariant to affine (translation + scaling) intensity change

Problem with intensity vectors

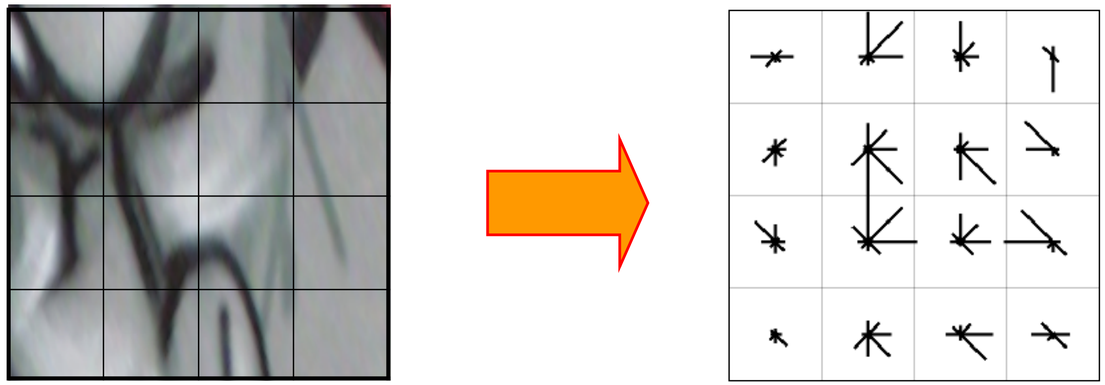
Matching score is can be effected by

- small deformations
- small changes in intensity

Feature descriptors: SIFT

Descriptor computation:

- Divide patch into 4x4 sub-patches
- Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
- Normalize to unit length
- Resulting descriptor: 4x4x8 = 128 dimensions



David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Feature descriptors: SIFT

Descriptor computation

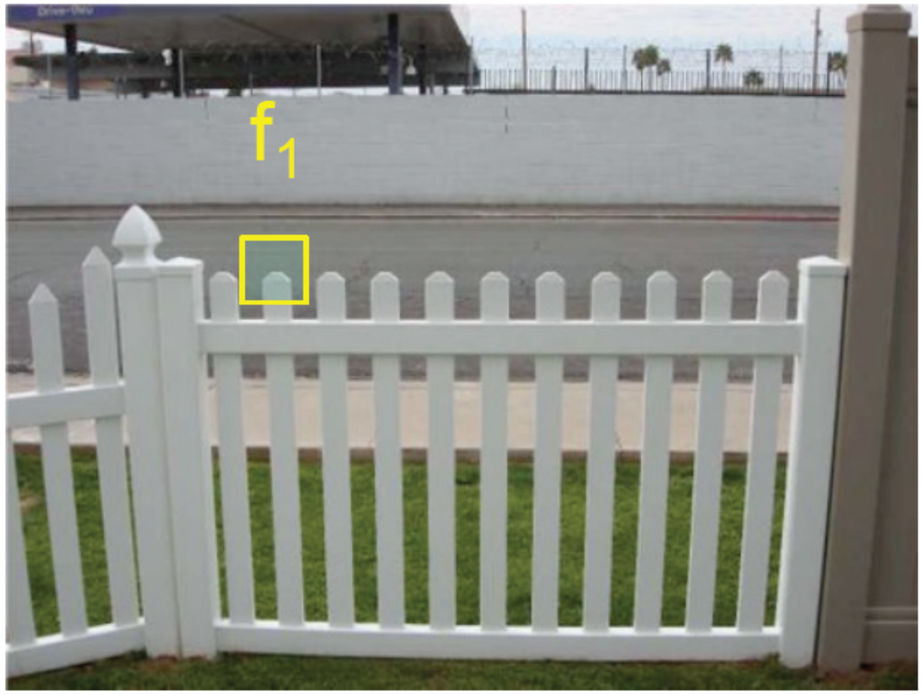
- Divide patch into 4x4 sub-patches
- Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
- Normalize to unit length
- Resulting descriptor: 4x4x8 = 128 dimensions

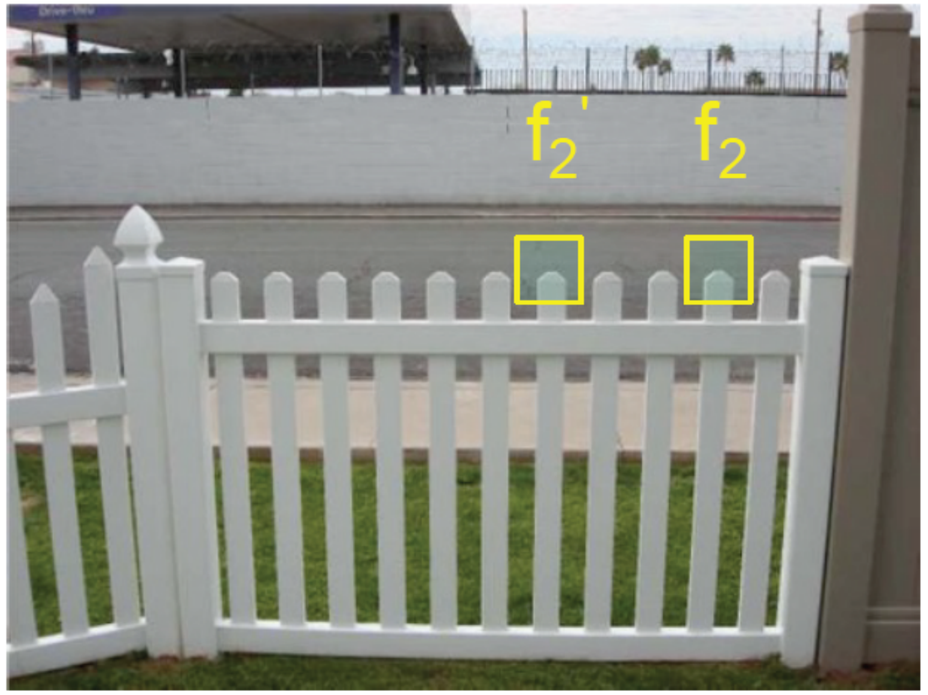
Advantage over raw vectors of pixel values

- Gradients less sensitive to illumination change
- Pooling of gradients over the sub-patches achieves robustness to small shifts, but still preserves some spatial information

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Problem: Ambiguous matches





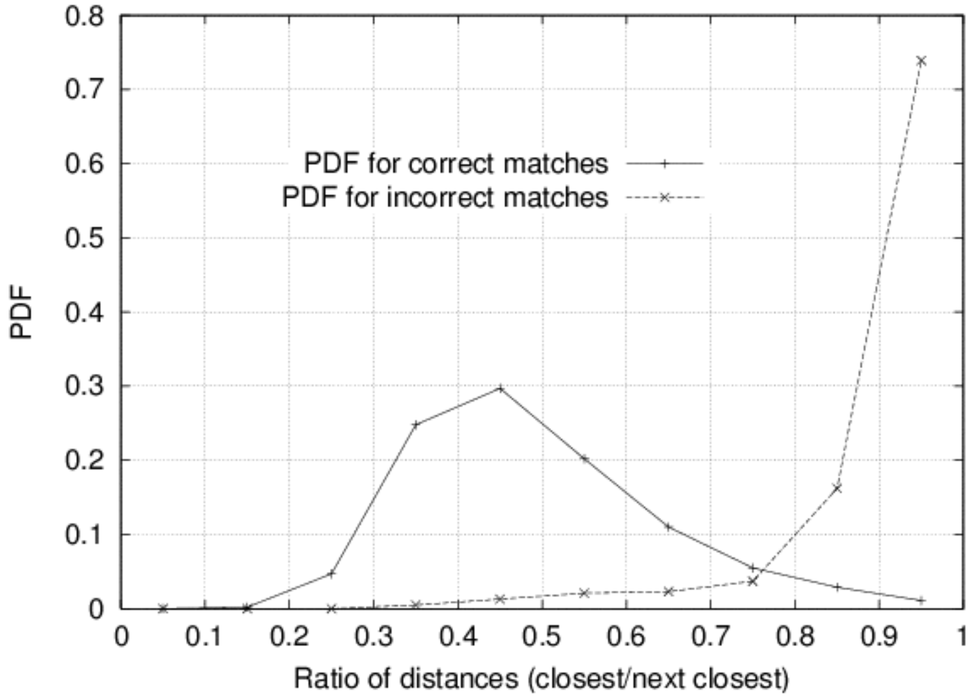
Source: Y. Furukawa

Rejection of unreliable matches

How can we tell which matches are more reliable?

Heuristic: compare distance of nearest neighbor to that of second nearest neighbor

- Ratio of closest to second-closest distance will be *high* for features that are *not* distinctive



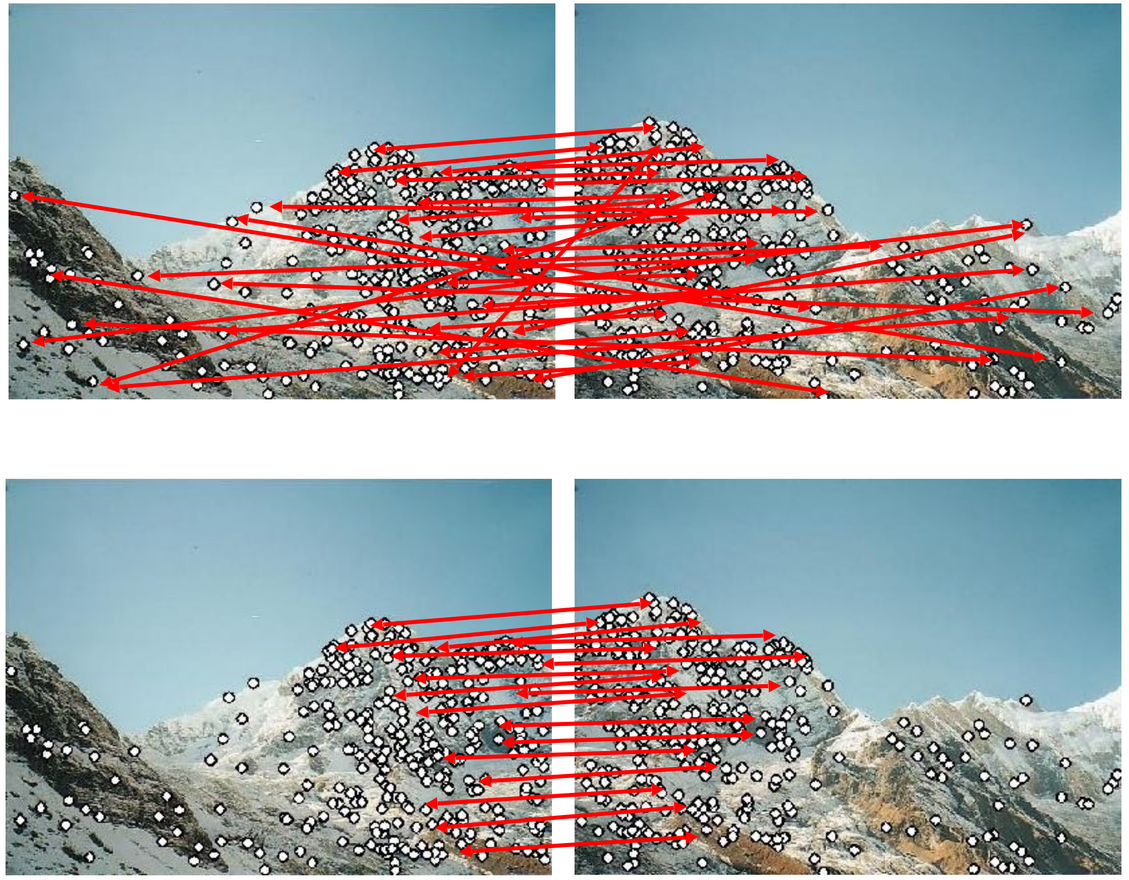
Threshold of 0.8 provides good separation

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

Run SIFT matching demo
(Also turn the ratio test on / off)

Consensus building

Given a number of putative matches how we find a set of consistent ones?

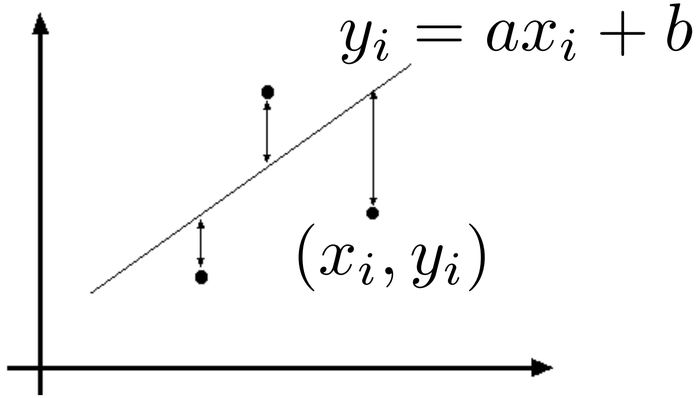


Least-squares line fitting

Data: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

Line equation: $y_i = ax_i + b$

$$\min_{a,b} \sum_i (ax_i + b - y_i)^2$$

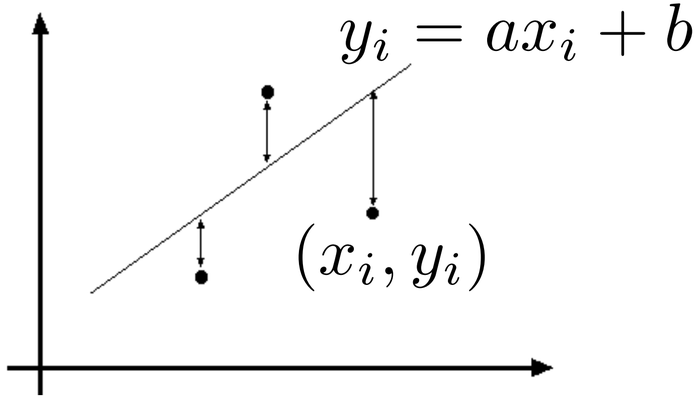


Least-squares line fitting

Data: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

Line equation: $y_i = ax_i + b$

$$\min_{a,b} \sum_i (ax_i + b - y_i)^2$$



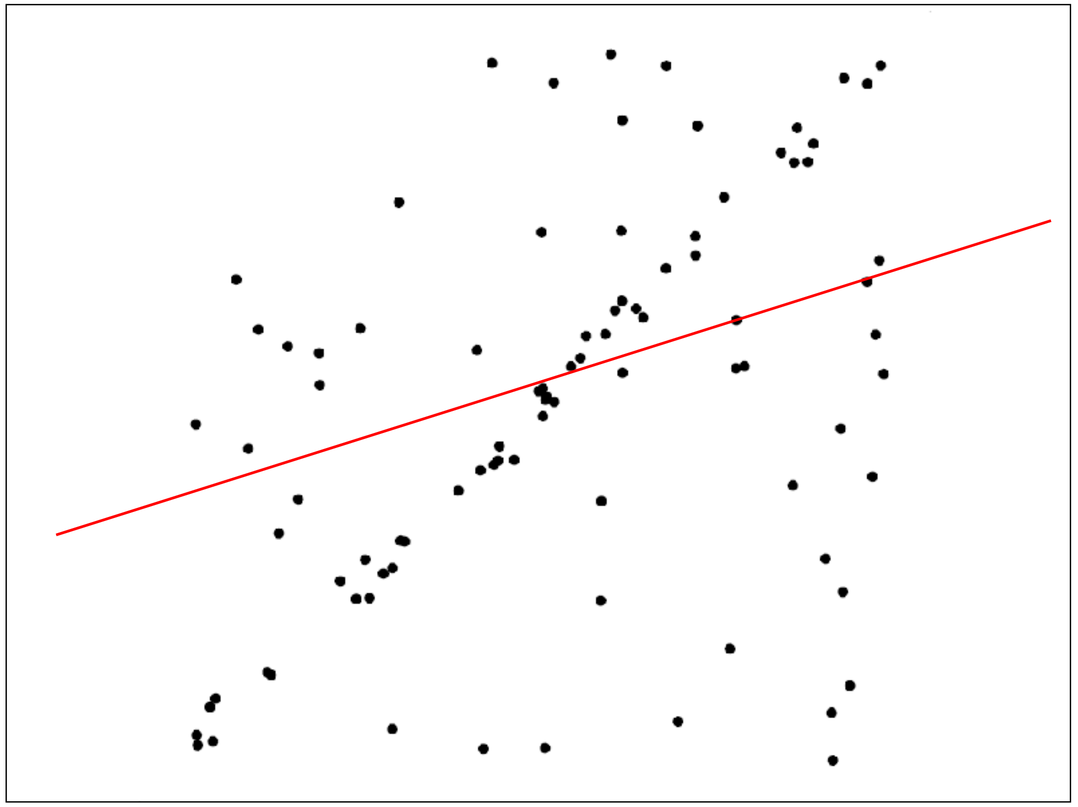
$$\mathbf{A} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \ldots & \ldots \\ 1 & x_n \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} b \\ a \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \ldots \\ y_n \end{bmatrix}$$

$$\mathbf{Ax} = \mathbf{y}$$

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

Problem: Outliers

$$\min_{a,b} \sum_i (ax_i + b - y_i)^2$$



Least-squares fit

Source: R. Raguram

RANSAC

Random **Sa**mple **C**onsensus

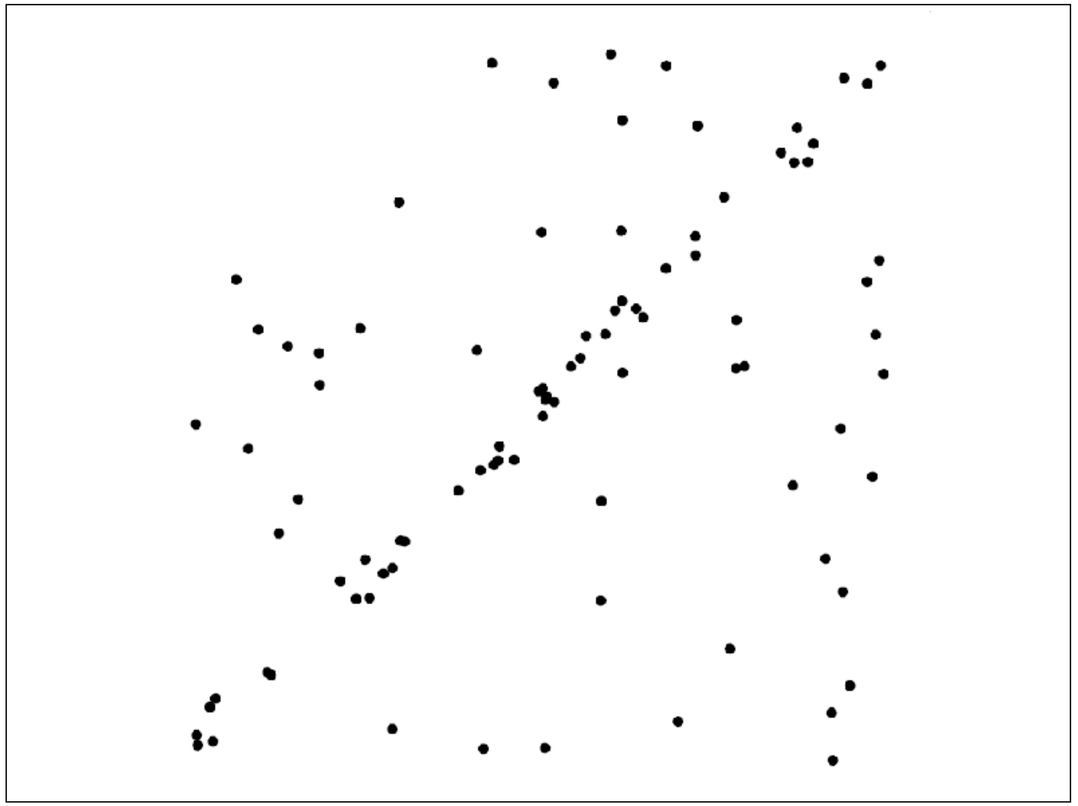
- Choose a small subset of points uniformly at random
- Fit a model to that subset
- Find all remaining points that are “close” to the model and reject the rest as outliers
- Do this many times and choose the best model

For rigid transformation, we can estimate the parameters of the transformation, e.g., rotation angle, scaling, translation, etc, from putative correspondence matches

Lets see how RANSAC works for a simple example

M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

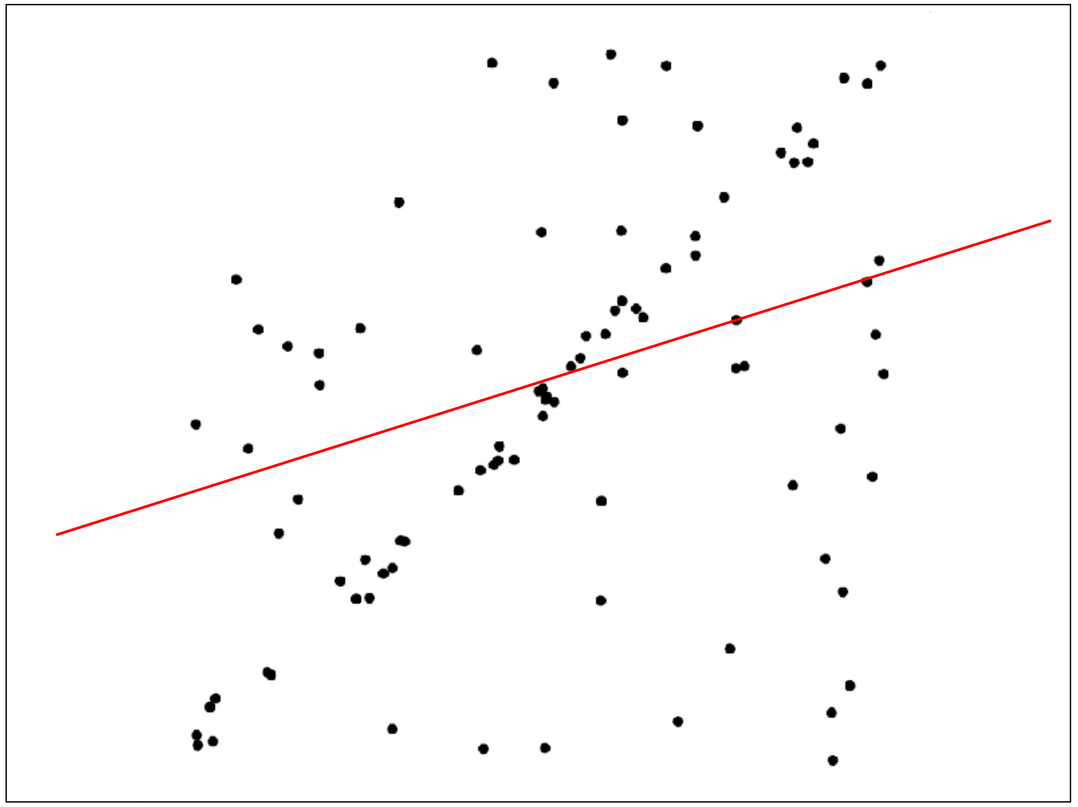
RANSAC for line fitting example



Source: R. Raguram

RANSAC for line fitting example

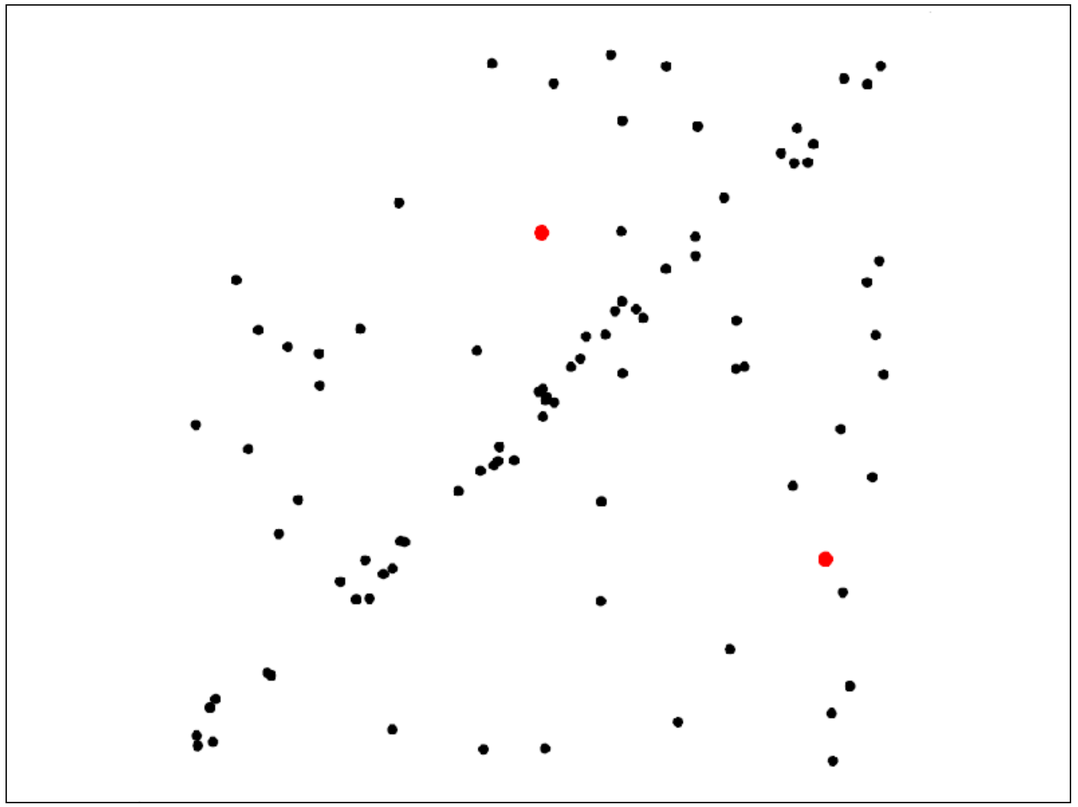
$$\min_{a,b} \sum_i (ax_i + b - y_i)^2$$



Least-squares fit

Source: R. Raguram

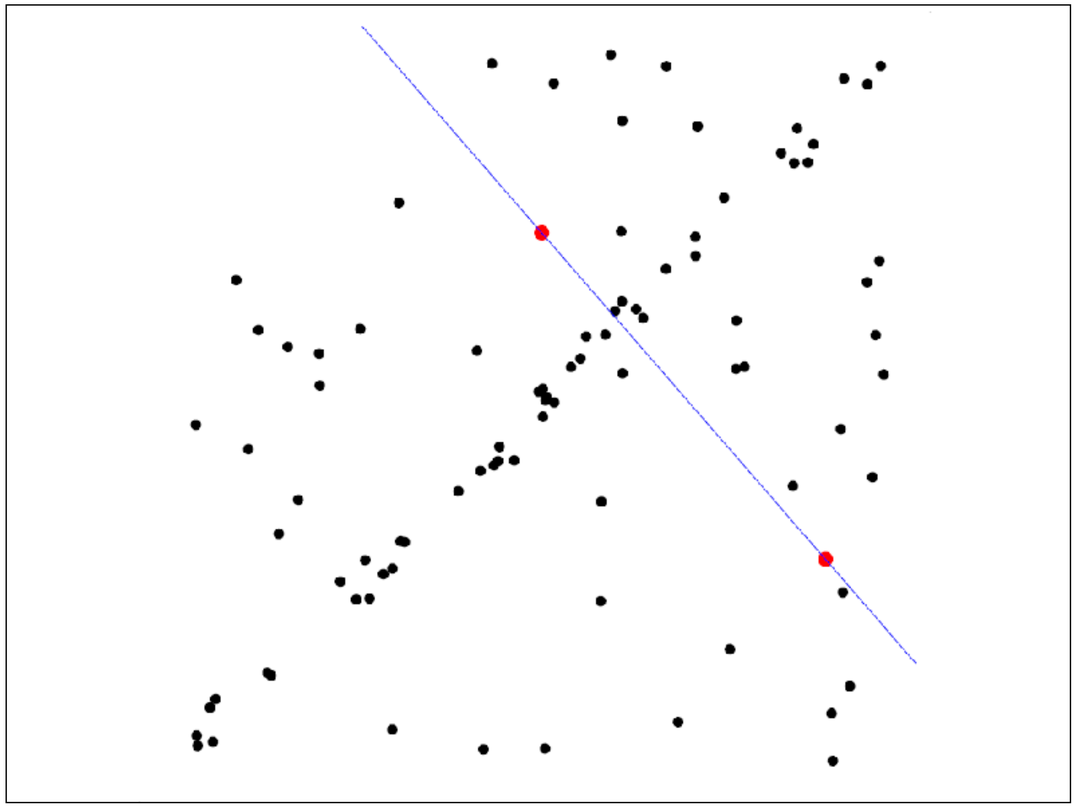
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

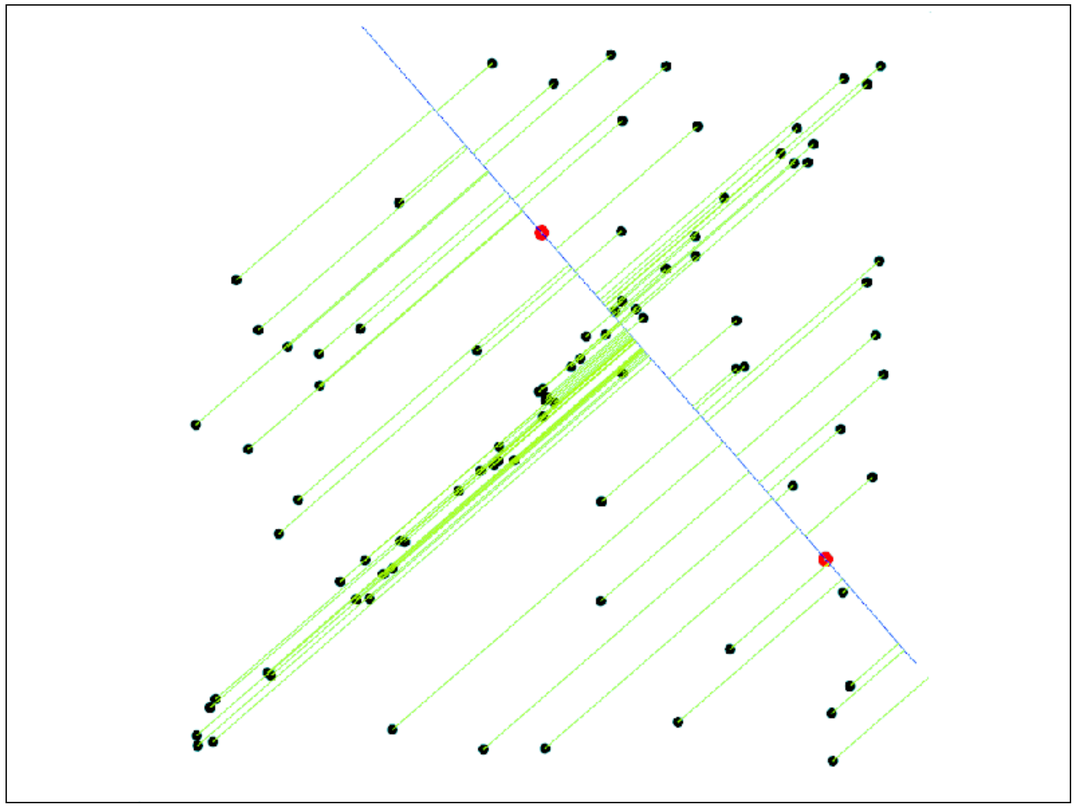
RANSAC for line fitting example



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

Source: R. Raguram

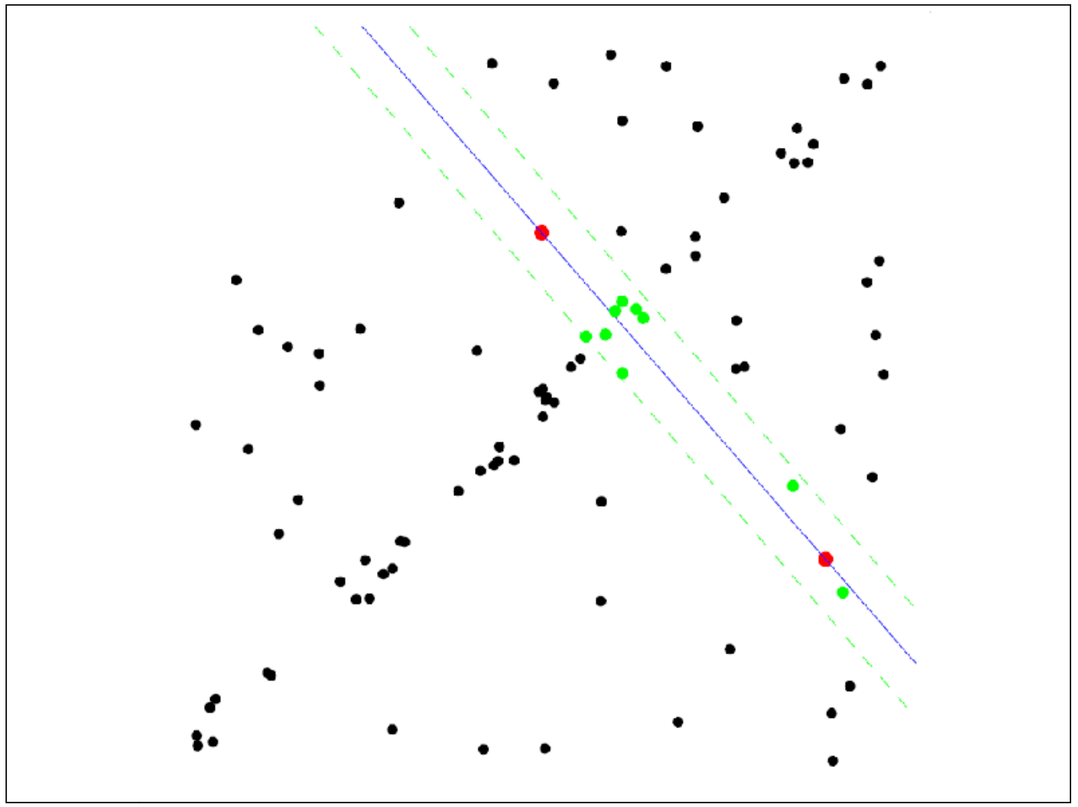
RANSAC for line fitting example



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

Source: R. Raguram

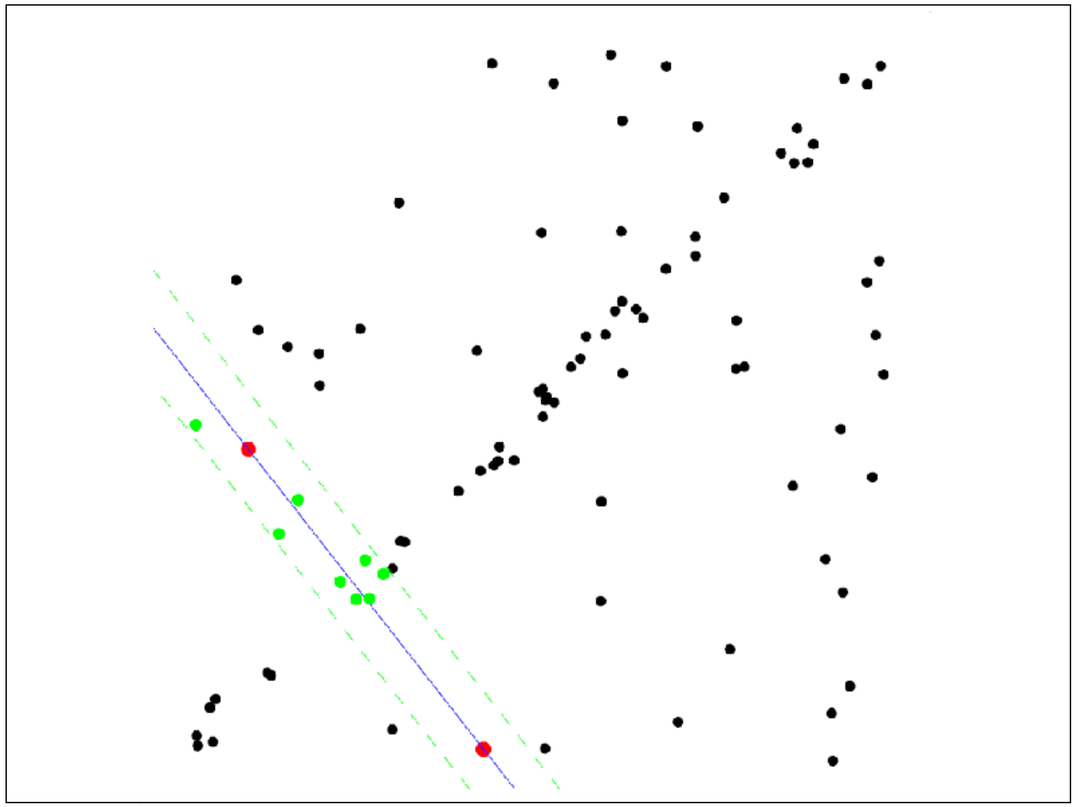
RANSAC for line fitting example



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

Source: R. Raguram

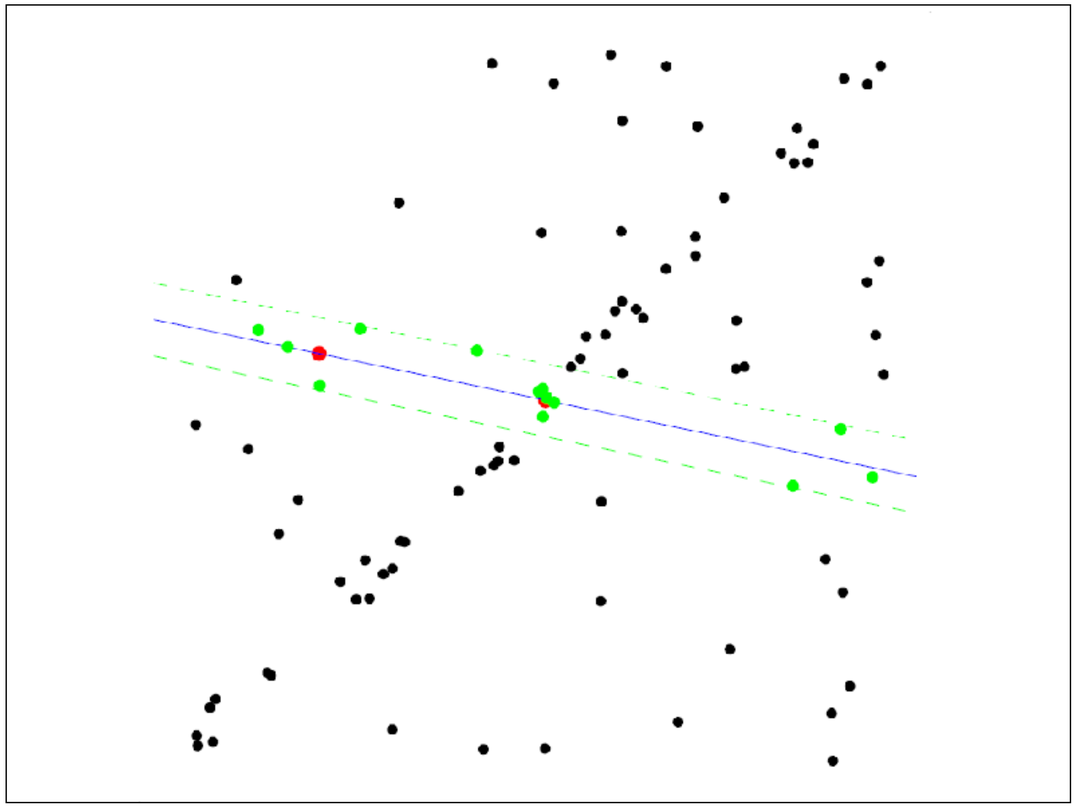
RANSAC for line fitting example



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

RANSAC for line fitting example

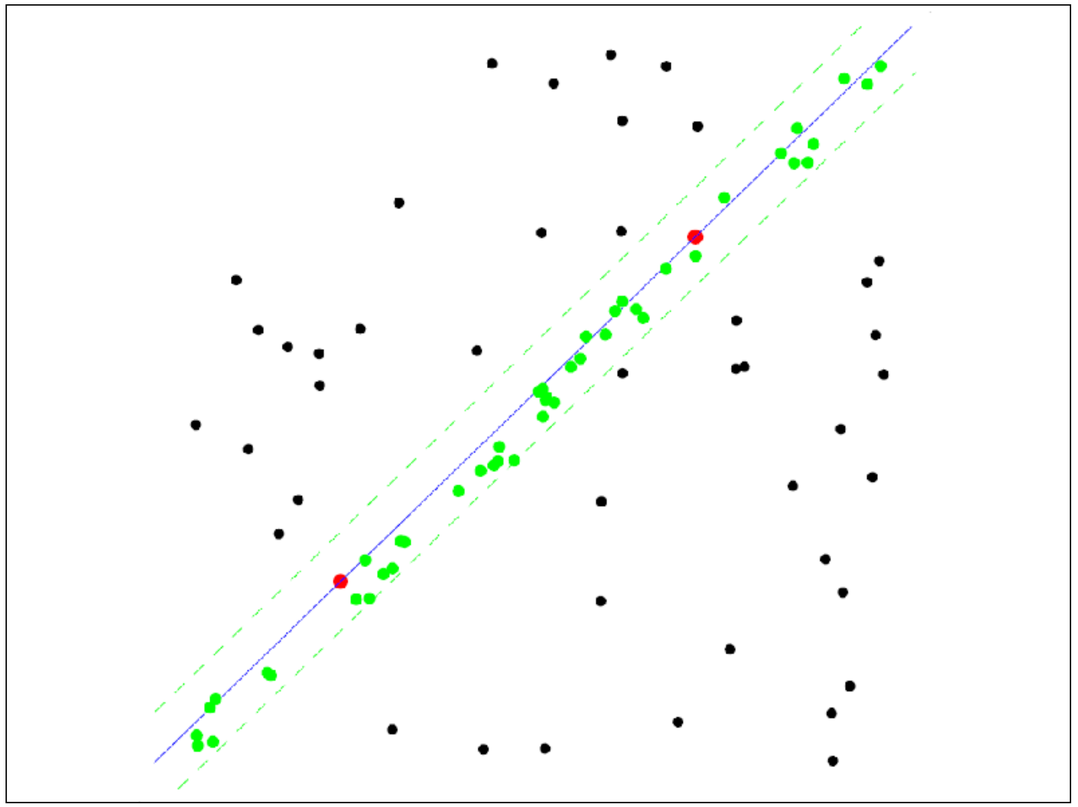


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

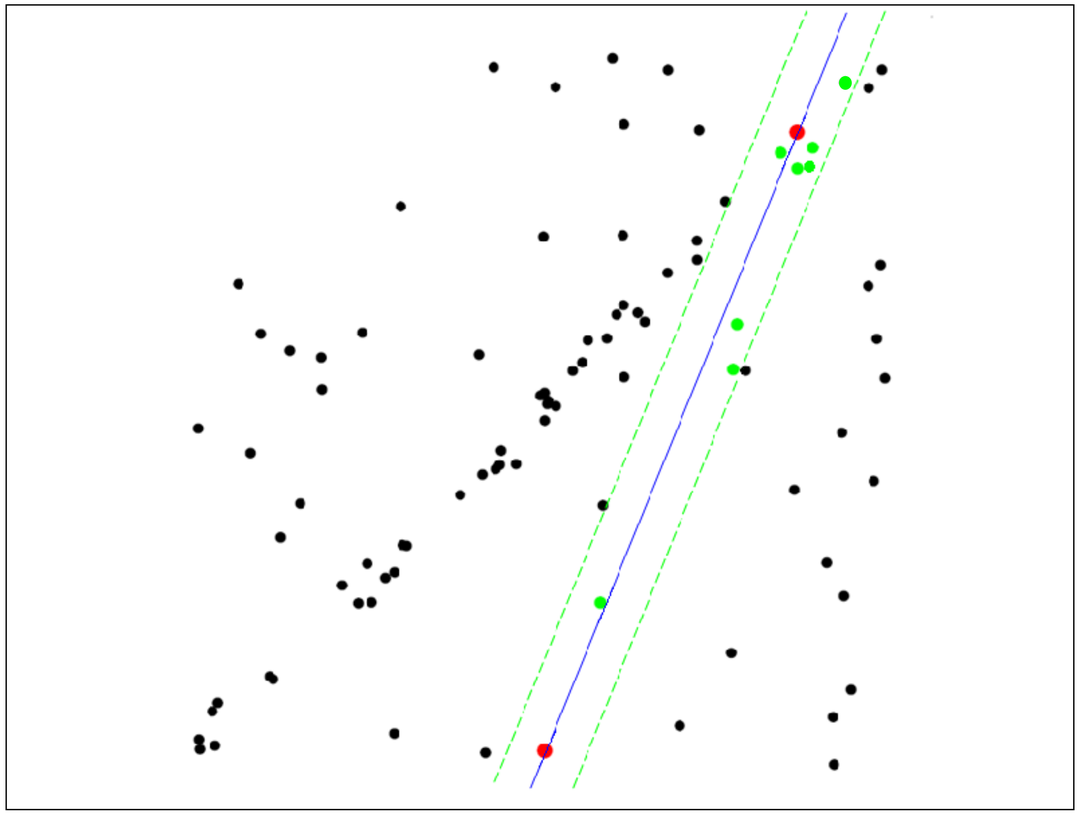
Uncontaminated sample



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 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

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 and the number of inliers is higher than the *previous best*, accept the line and refit using all inliers.

How robust is RANSAC?

Suppose a fraction of points p are outliers

- What is the probability that RANSAC will fail in 1 iteration?

- What is the probability that RANSAC will fail in T iterations?

Suppose p=0.5. How many iterations do we need for the probability of success to be > 0.99?

How robust is RANSAC?

Suppose a fraction of points p are outliers

- What is the probability that RANSAC will fail in 1 iteration?
  - $q = 1 - (1-p)^2$

- What is the probability that RANSAC will fail in T iterations?
  - q^T

Suppose p=0.5. How many iterations do we need for the probability of success to be > 0.99?

- We need: $1 - q^T > 0.99$

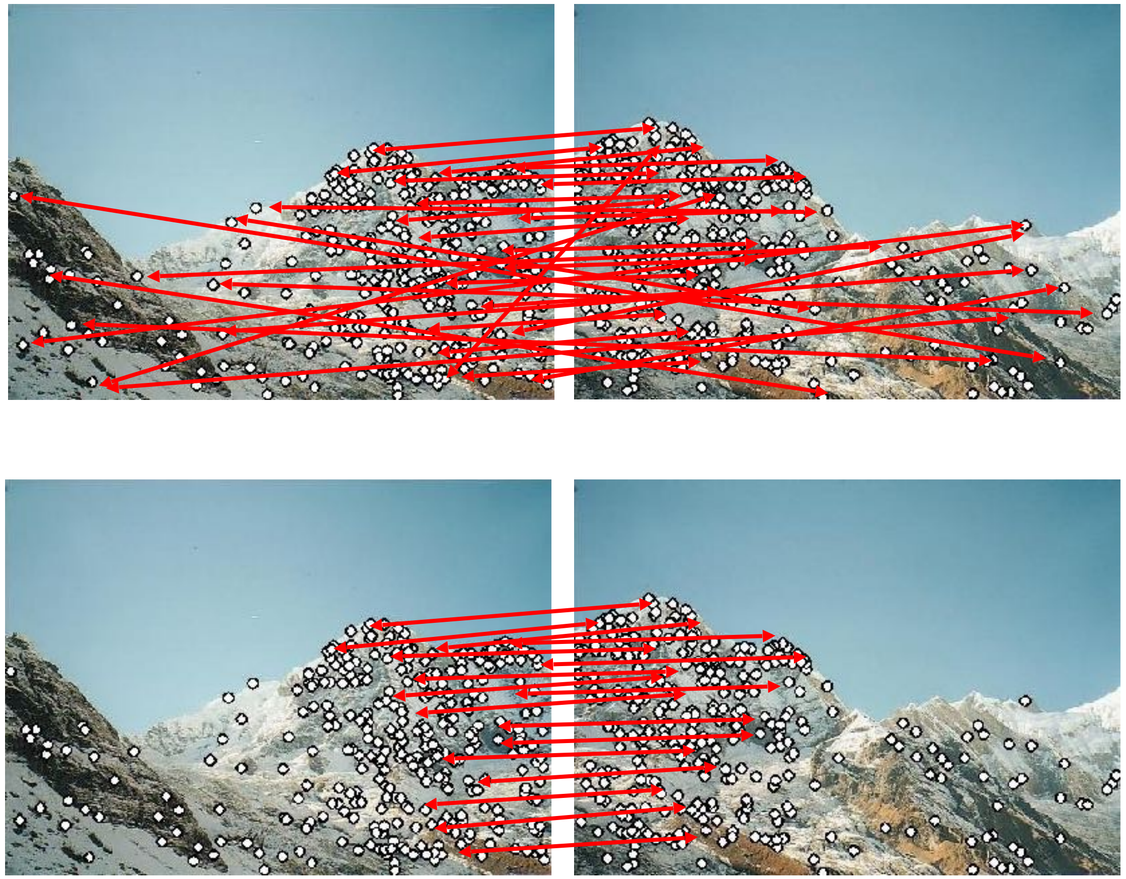
  ⇨ $1 - (1-(1/4))^T > 0.99$

  ⇨ $T > \log_2(0.01)/\log_2(0.75) \sim 17$

0.999, T~24
0.9999, T~32
0.99999, T~40

RANSAC for image matching

Select inliers based on their agreement with an underlying transformation



How do we represent image transformations?

Families of transformation

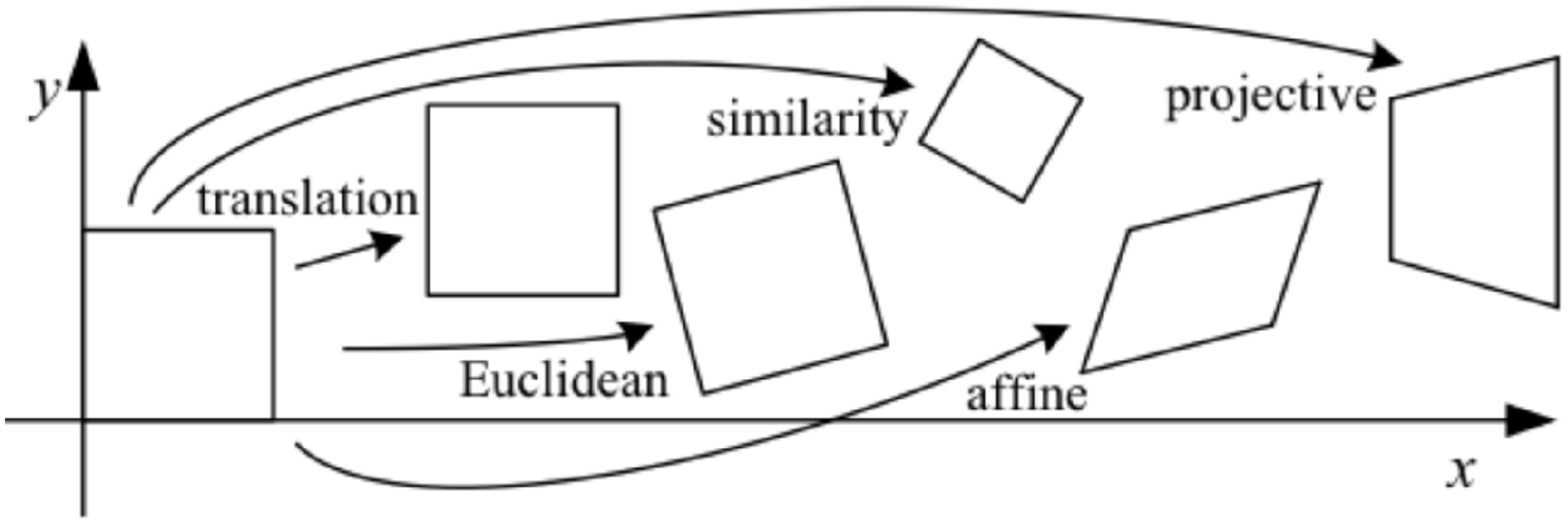


Figure 2.4 Basic set of 2D planar transformations.

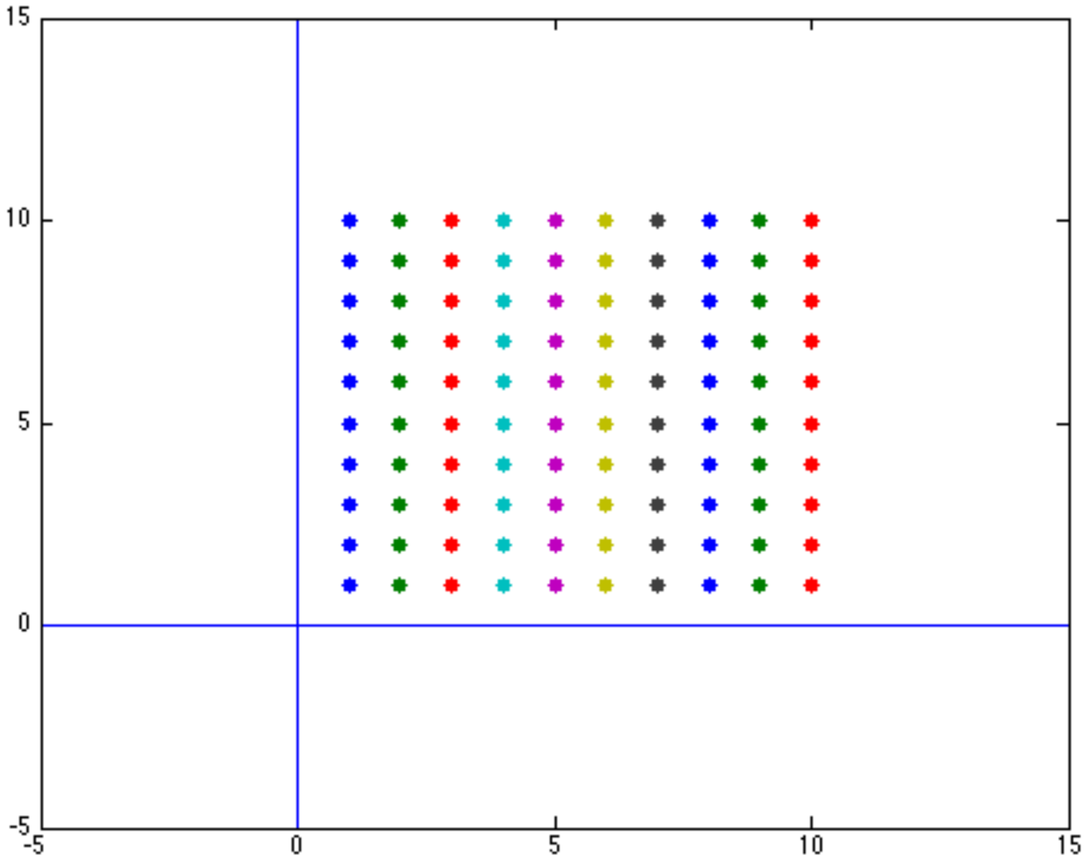
From Computer Vision: Algorithms and Applications, by Rick Szeliski

How do we move pixels?

Think about moving coordinates, not pixels.

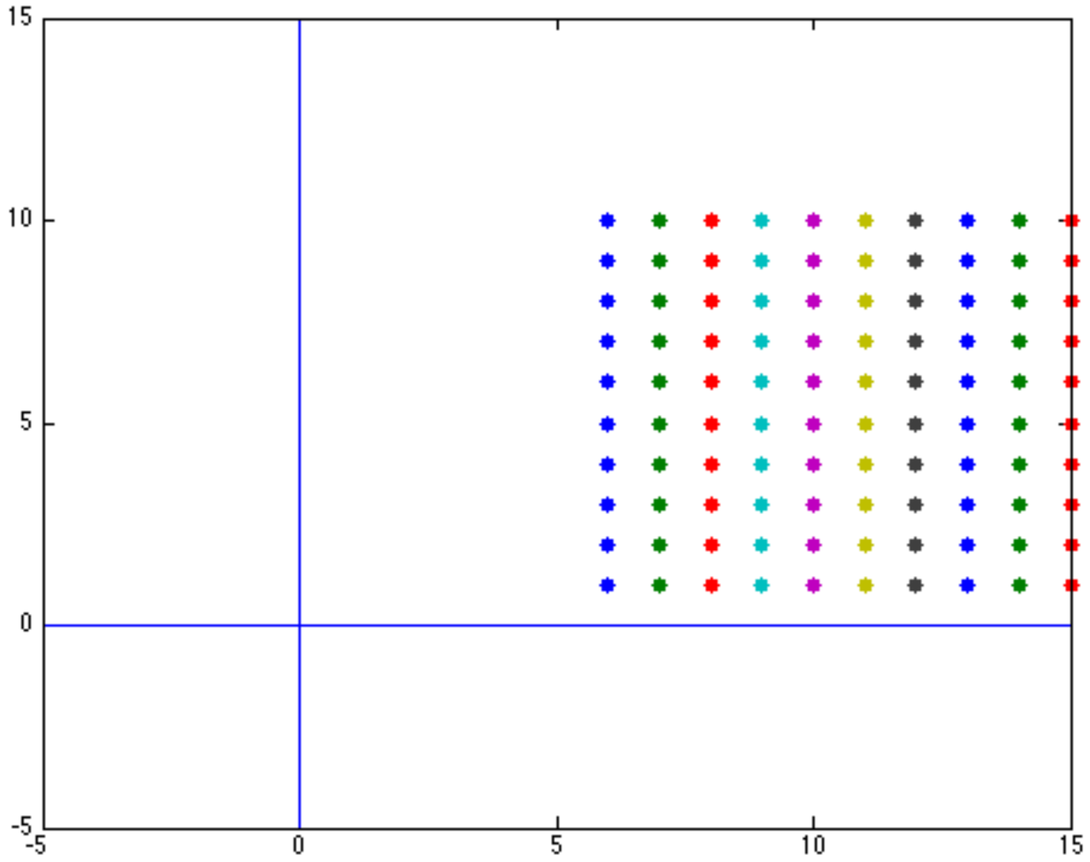
Points in 2D

```
figure(1);
[ox oy]=meshgrid(1:10,1:10);
plot(ox,oy,'.');
line([0 0],[-5 15]);
line([-5 15],[0 0]);
axis([-5 15 -5 15]);
```



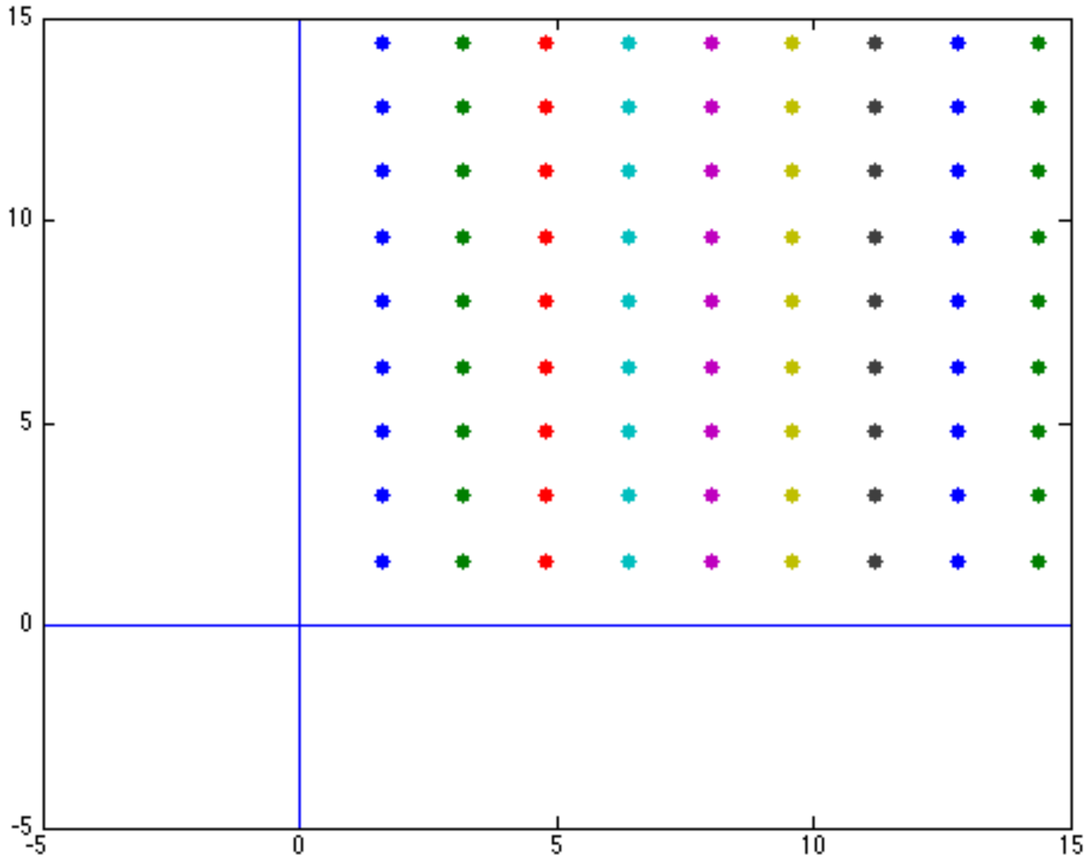
Translation

```
figure(2);
nx = ox + 5;
ny = oy;

plot(nx,ny,'.');
line([0 0],[-5 15]);
line([-5 15],[0 0]);
axis([-5 15 -5 15]);
```



Scaling

```
figure(3);
oxy = [ox(:)'; oy(:)'];
A = [1.6 0; 0 1.6];
nxy = A*oxy;

nx = nxy(1,:);
ny = nxy(2,:);
nx = reshape(nx, [10 10]);
ny = reshape(ny, [10 10]);
plot(nx,ny,'.');
line([0 0],[-5 15]);
line([-5 15],[0 0]);
axis([-5 15 -5 15]);
```



Uniform scaling:

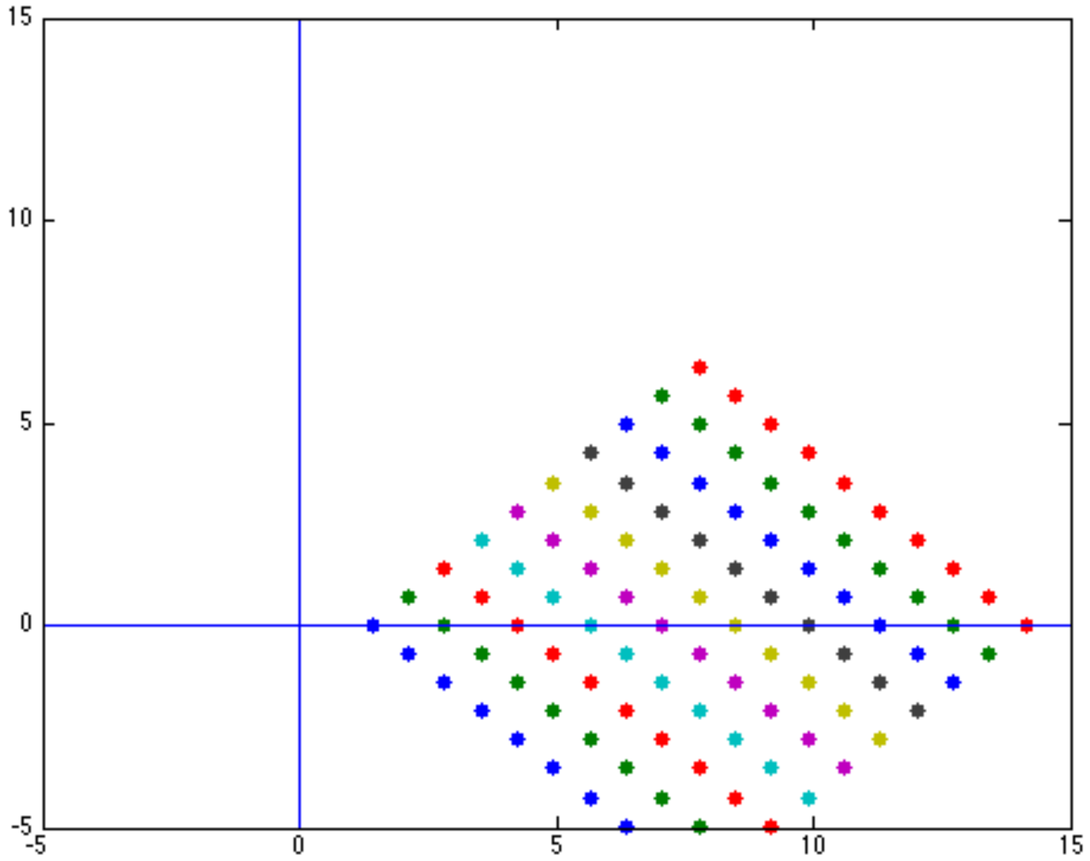
$$\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

Scaling in x:

$$\begin{bmatrix} s_x & 0 \\ 0 & 1 \end{bmatrix}$$

Rotation

```
figure(4);
oxy = [ox(:)'; oy(:)'];
A = [.707 .707;.707 -.707];
nxy = A*oxy;

nx = nxy(1,:);
ny = nxy(2,:);
nx = reshape(nx,[10 10]);
ny = reshape(ny,[10 10]);
plot(nx,ny,'.');
line([0 0],[-5 15]);
line([-5 15],[0 0]);
axis([-5 15 -5 15]);
```

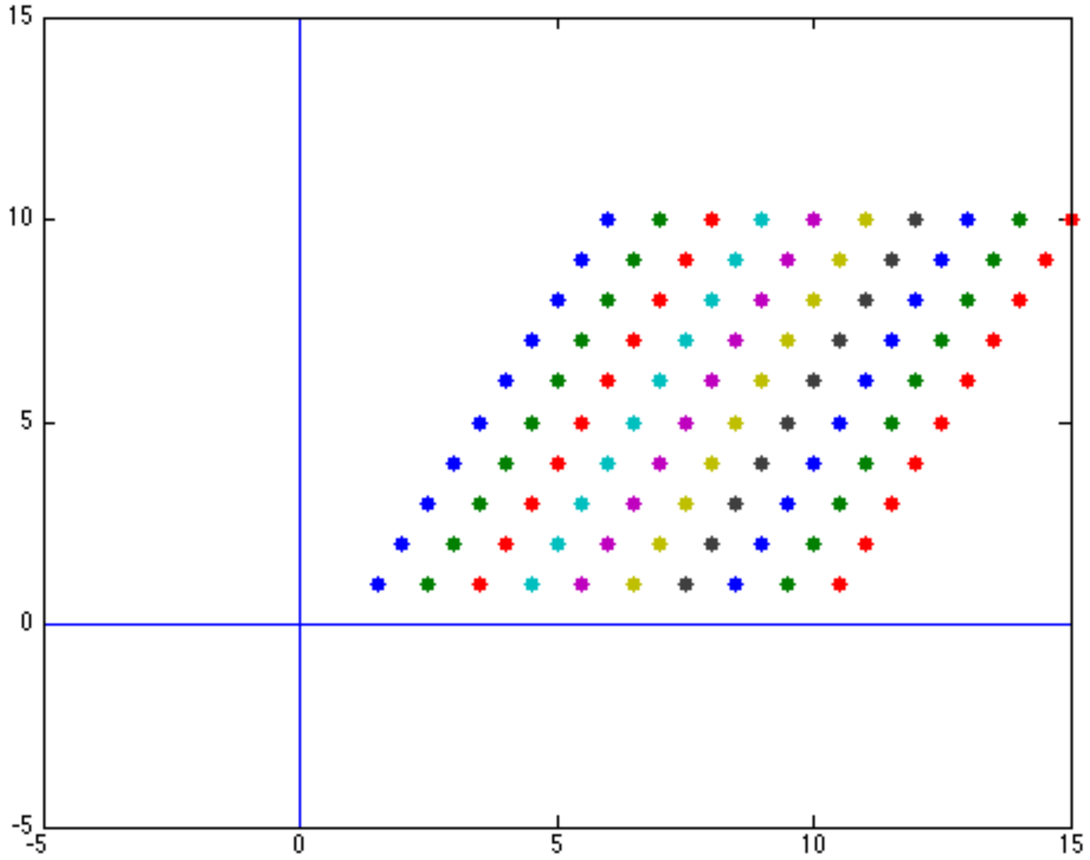


Rotation by θ radians:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Shear

```
figure(5);
oxy = [ox(:)'; oy(:)'];
A = [1 0.5; 0 1];
nxy = A*oxy;

nx = nxy(1,:);
ny = nxy(2,:);
nx = reshape(nx,[10 10]);
ny = reshape(ny,[10 10]);
plot(nx,ny,'.');
line([0 0],[-5 15]);
line([-5 15],[0 0]);
axis([-5 15 -5 15]);
```

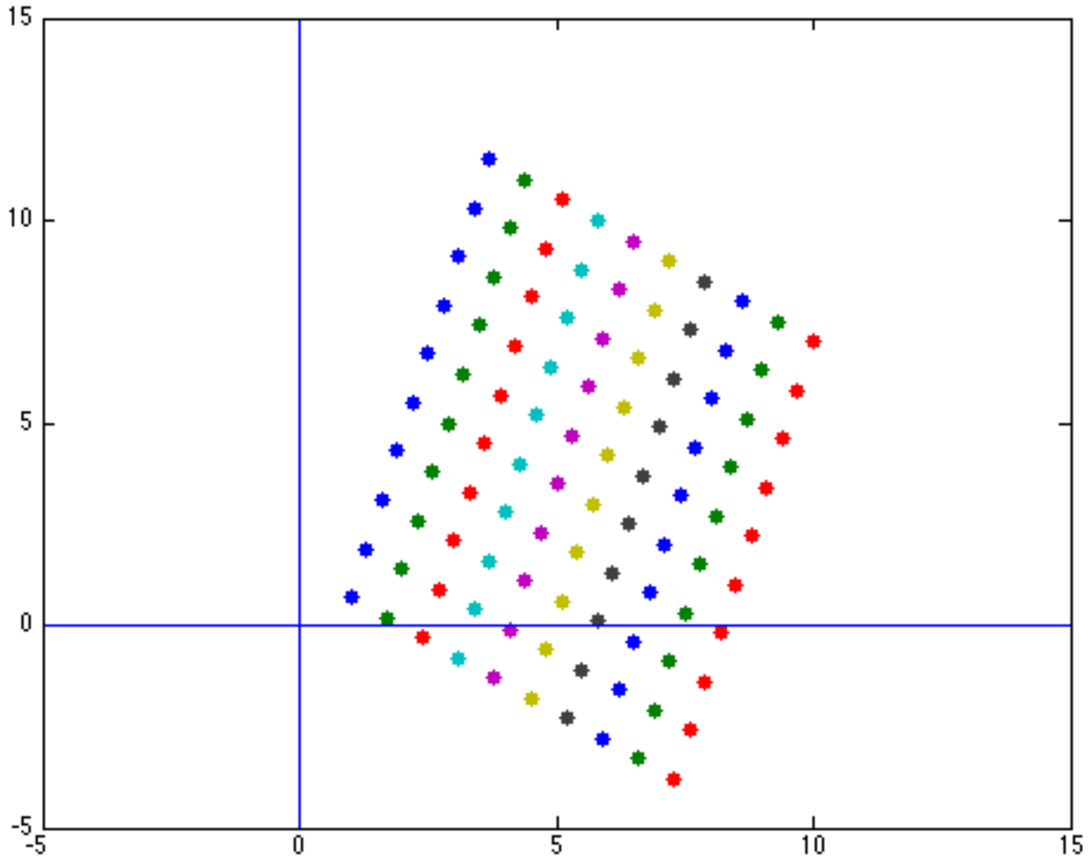


Shearing in x:

$$\begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix}$$

Arbitrary linear transformation

```
figure(6);
oxy = [ox(:)'; oy(:)'];
A = [0.7 0.3; -0.5 1.2];
nxy = A*oxy;

nx = nxy(1,:);
ny = nxy(2,:);
nx = reshape(nx,[10 10]);
ny = reshape(ny,[10 10]);
plot(nx,ny,'.');
line([0 0],[-5 15]);
line([-5 15],[0 0]);
axis([-5 15 -5 15]);
```



Arbitrary linear transformation:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Families of linear transforms

Identity:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Uniform scaling:

$$\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

Scaling in x:

$$\begin{bmatrix} s_x & 0 \\ 0 & 1 \end{bmatrix}$$

Rotation by θ radians:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Shearing in x:

$$\begin{bmatrix} 1 & sh_x \\ 0 & 1 \end{bmatrix}$$

Arbitrary linear transformation:

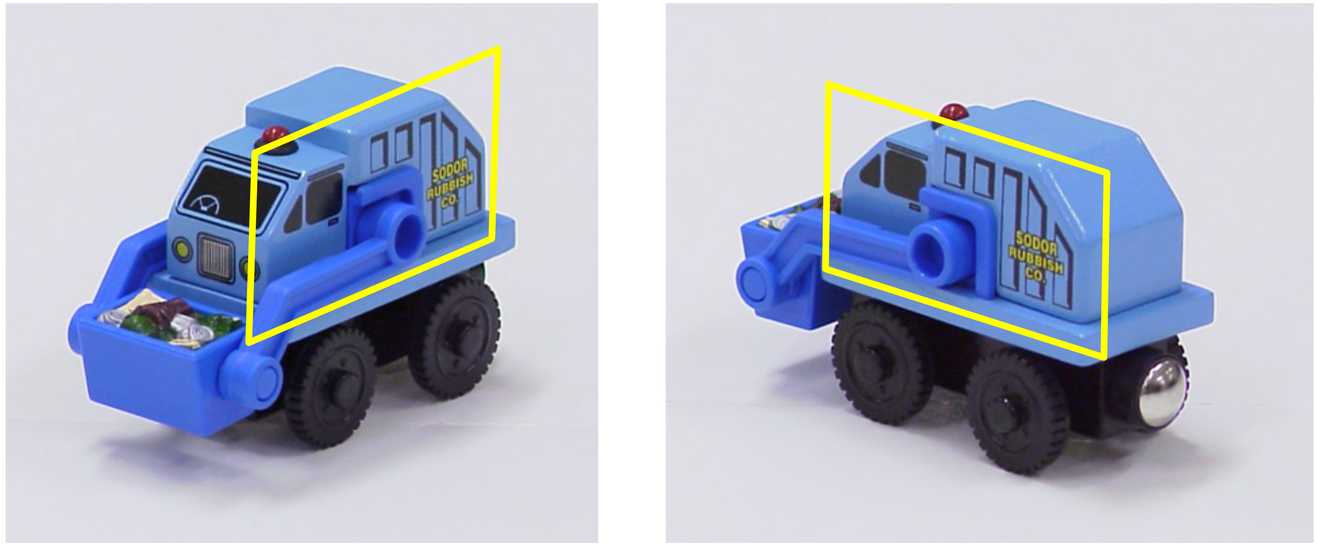
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Affine transformations

Affine = linear + translation $\mathbf{x} \rightarrow \mathbf{M}\mathbf{x} + \mathbf{t}$

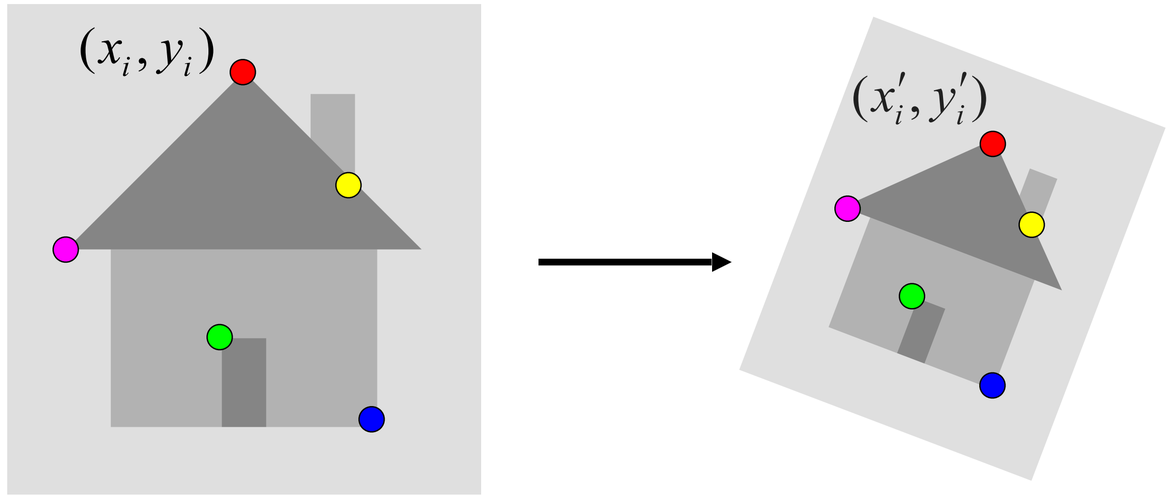
Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras

Can be used to initialize fitting for more complex models



Fitting an affine transformation

Assume we know the correspondences, how do we get the transformation?



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

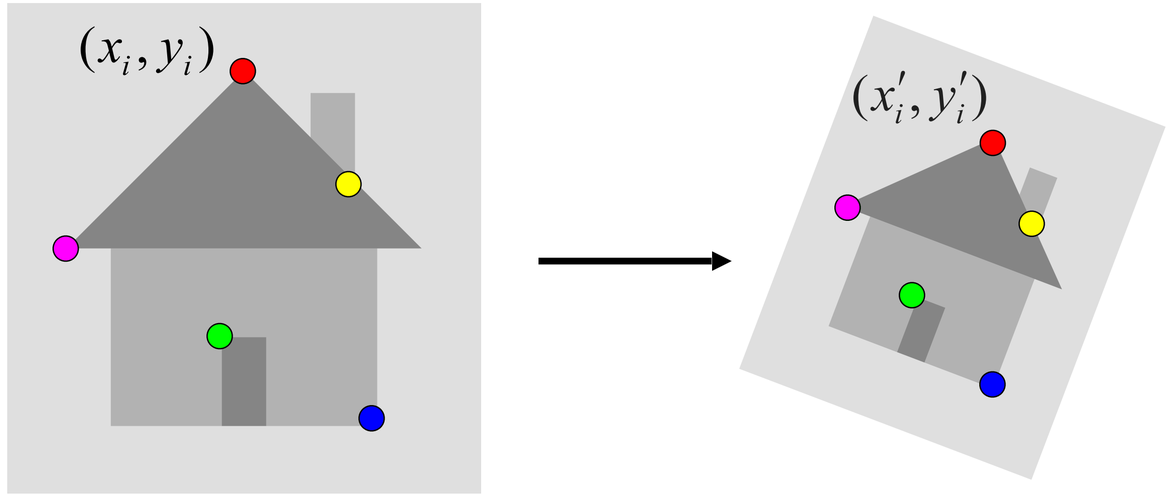
$$\mathbf{x}_i' = \mathbf{M}\mathbf{x}_i + \mathbf{t}$$

Want to find **M**, **t** to minimize

$$\sum_{i=1}^{n} || \mathbf{x}_i' - \mathbf{M}\mathbf{x}_i - \mathbf{t} ||^2$$

Fitting an affine transformation

Assume we know the correspondences, how do we get the transformation?



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

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

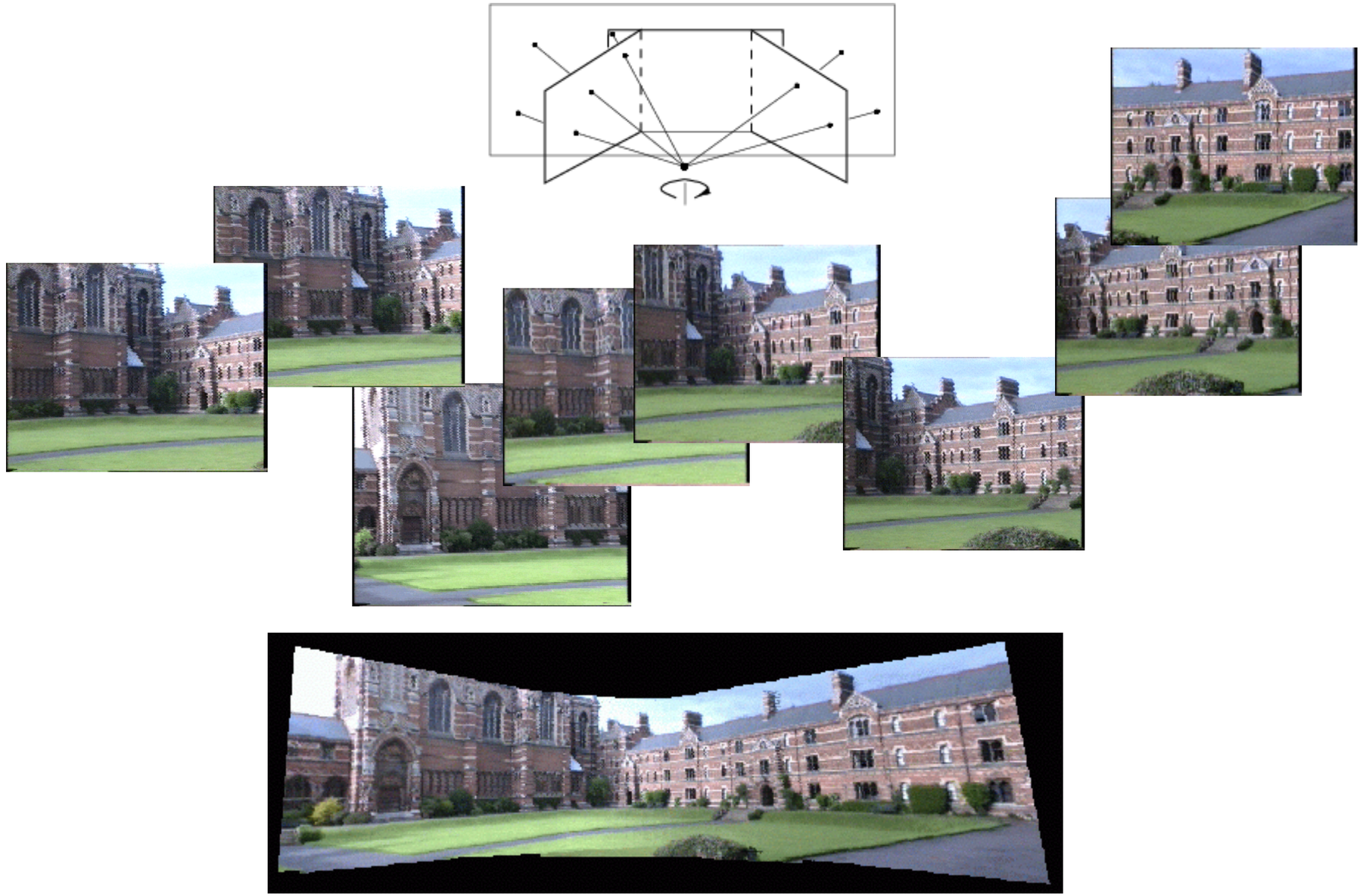
Fitting an affine transformation

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

Linear system with six unknowns

Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters

Application: Panorama stitching

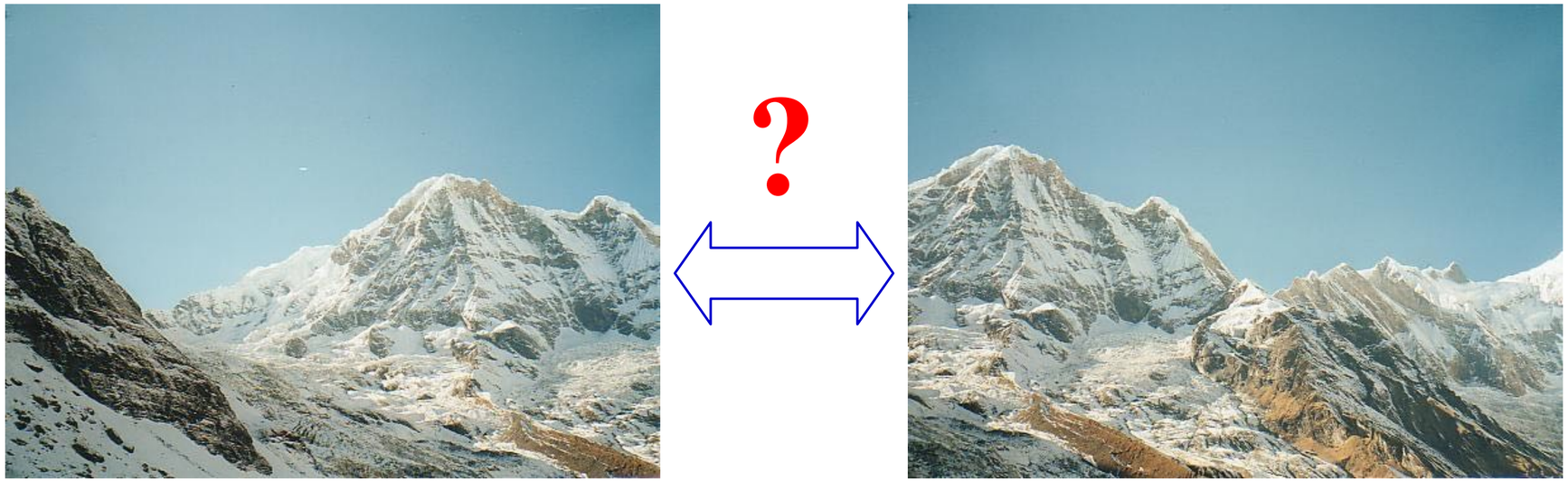


Source: Hartley & Zisserman

Panoramic stitching

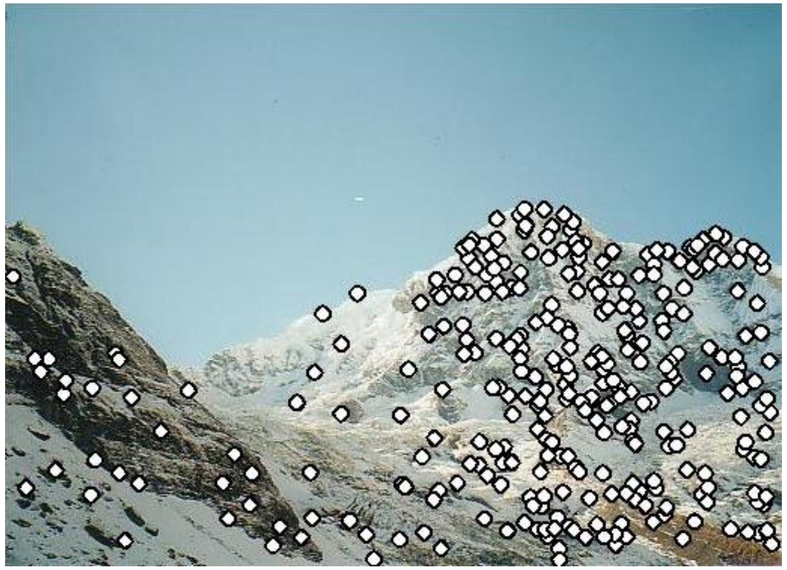
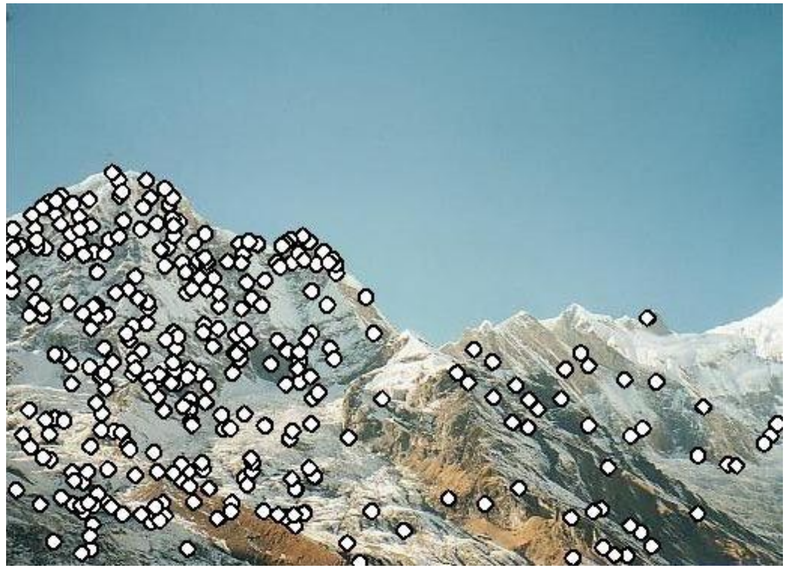
Approach

- Local feature matching
- RANSAC for alignment



Panoramic stitching

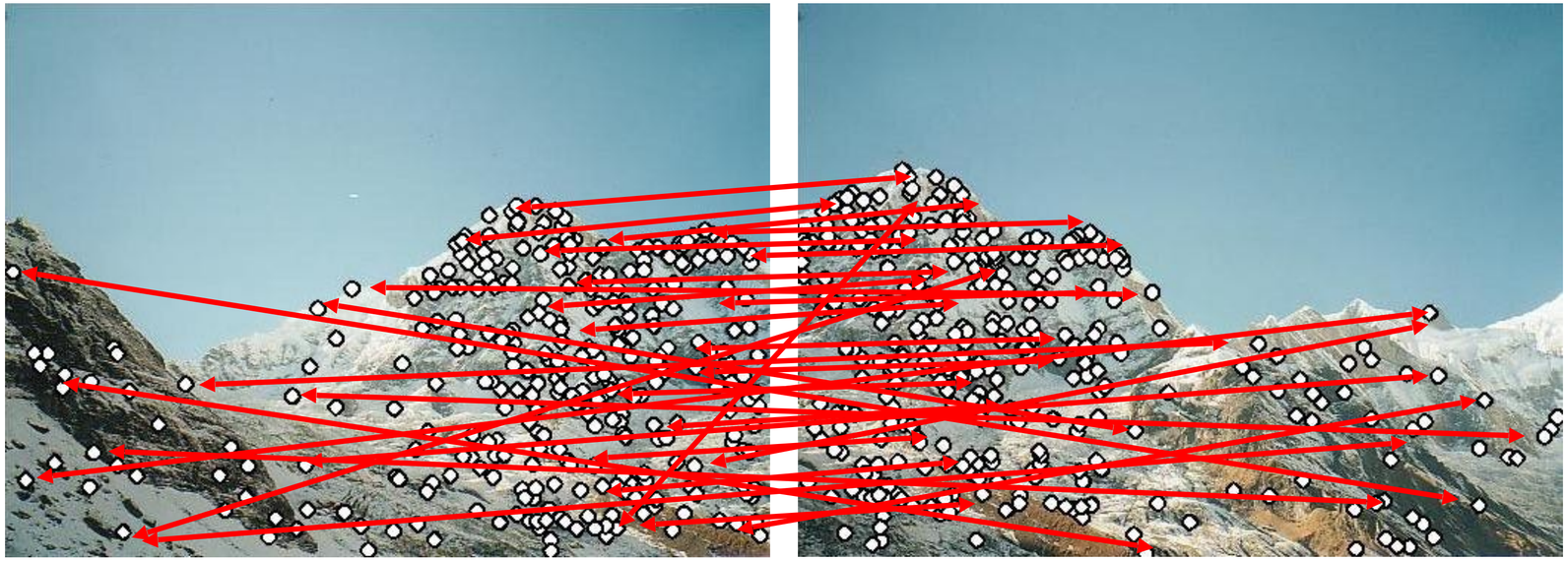
Panoramic stitching

Extract features

- corner/blob detector

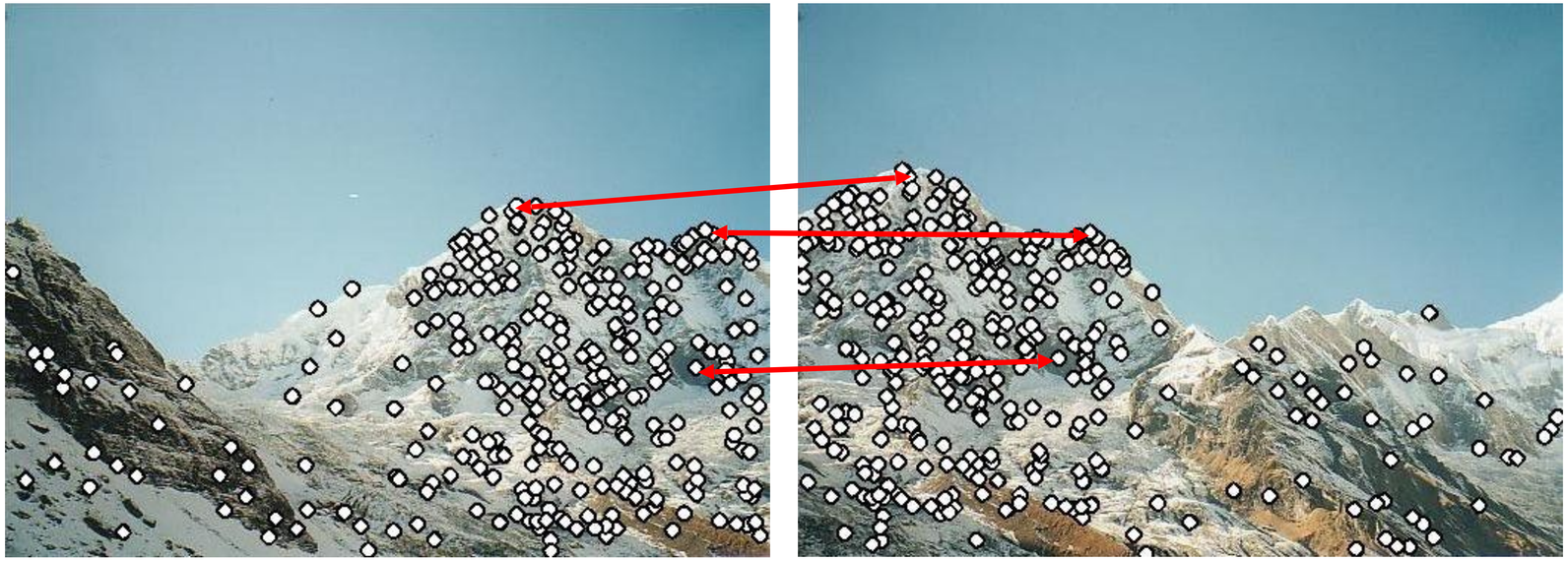
Panoramic stitching



Extract features

Compute *putative matches*

Panoramic stitching



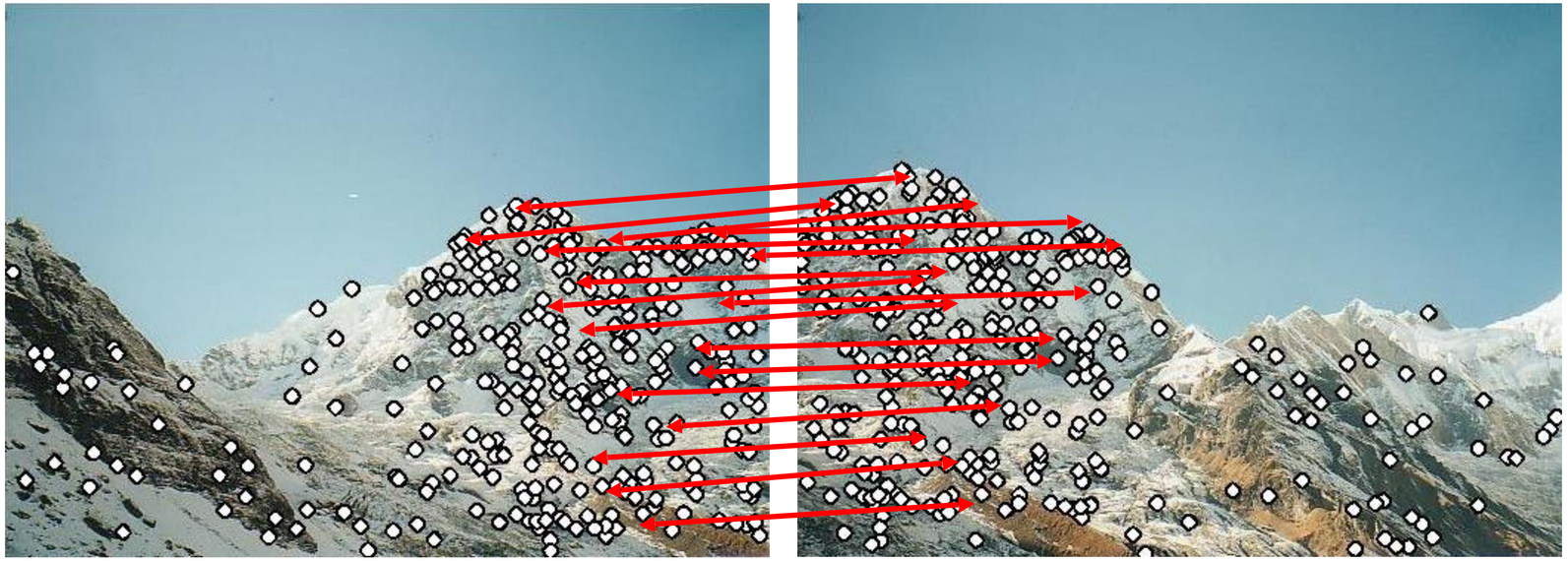
Extract features

Compute *putative matches*

Loop:

- *Hypothesize* transformation *T*

Panoramic stitching



Extract features

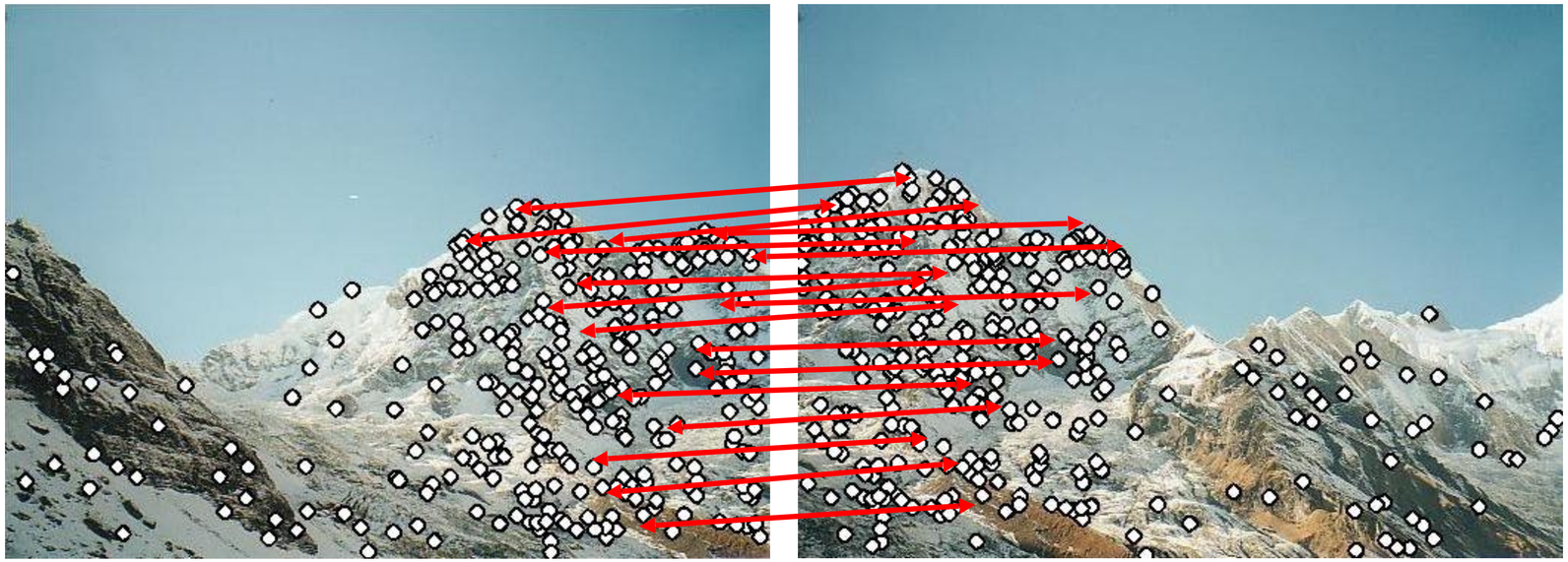
Compute *putative matches*

Loop:

- *Hypothesize* transformation *T*
- *Verify* transformation (search for other matches consistent with *T*)

Run SIFT matching + RANSAC demo

Final step: warping



T

Mechanics of transformation

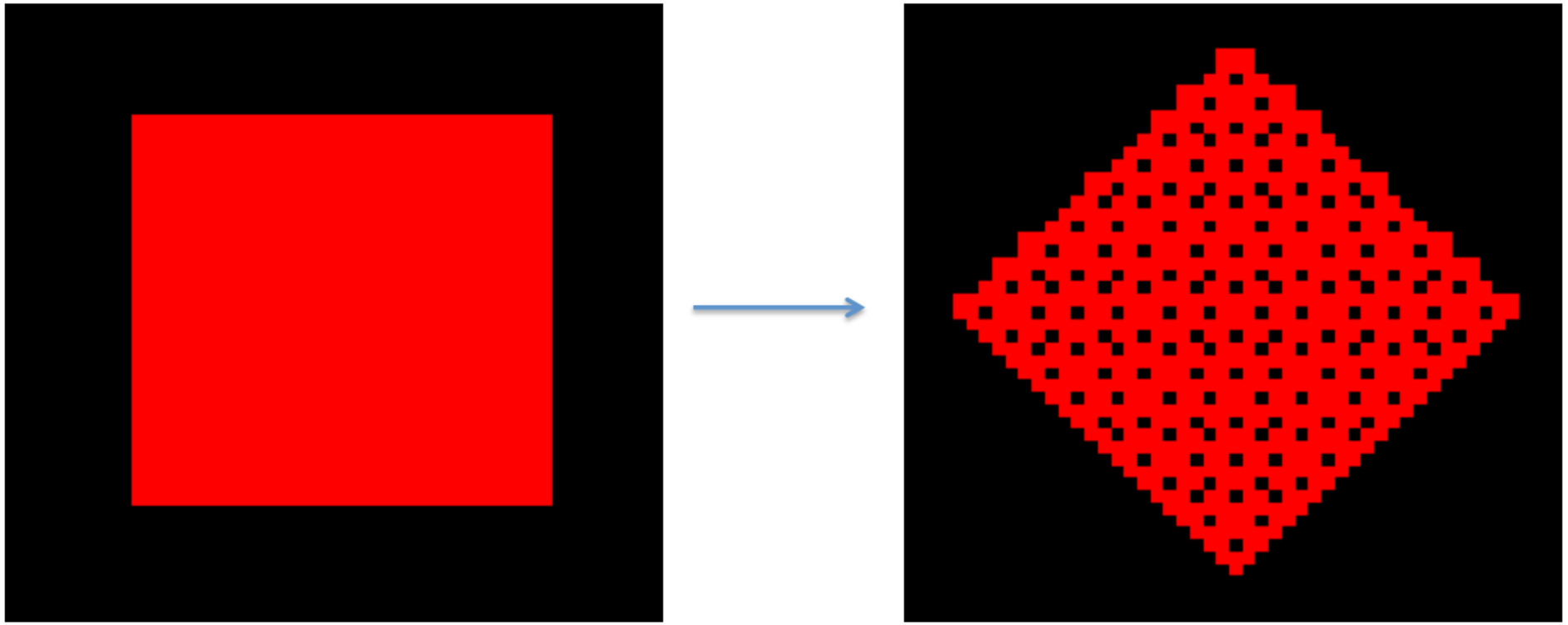
procedure *forwardWarp*($f, \boldsymbol{h}$, **out** g):

For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$

1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

Example of forward warp

Rotation by 45 degrees



Mechanics of transformation

procedure *inverseWarp*$(f, \boldsymbol{h}, \textbf{out}\ g)$:

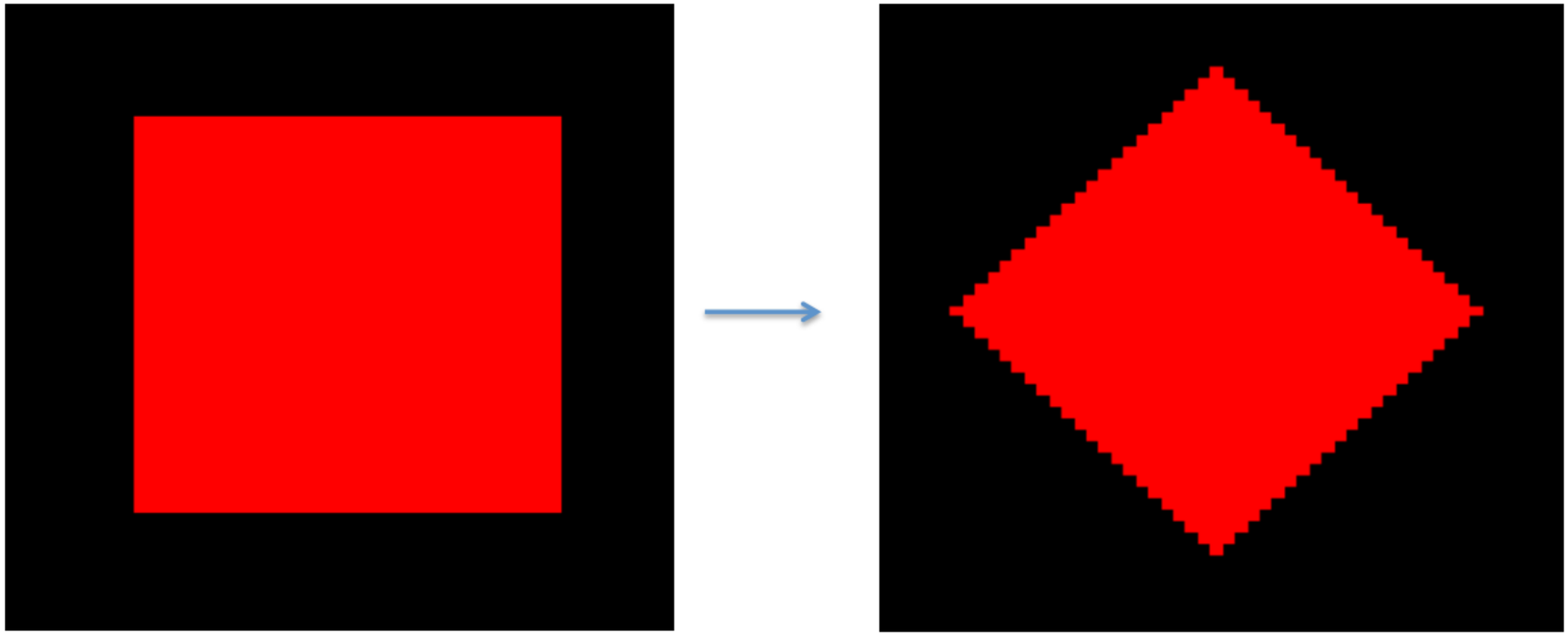
For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$

1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

From Computer Vision: Algorithms and Applications, by Rick Szeliski

Example of inverse warp

Rotation by 45 degrees



Putting it all together