Computer Vision

CS-E4850, 5 study credits

Lecturer: Juho Kannala

Lecture 5: Image alignment and object instance recognition

- Given two images of a planar scene (or from a rotating camera), find the parameters of a global geometric transformation that accounts for most true point correspondences between the images
- Given a query image and a database of object images, detect whether the objects are visible in the query image
- Reading:
  - Szeliski's book, Sections 8.1.1 - 8.1.4 in 2nd edition
  - Chapter 4 in Hartley & Zisserman
  - Lowe's SIFT paper: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

Acknowledgement: many slides from Svetlana Lazebnik, Derek Hoiem, Kristen Grauman, and others (detailed credits on individual slides)

Source: S. Lazebnik

Image alignment

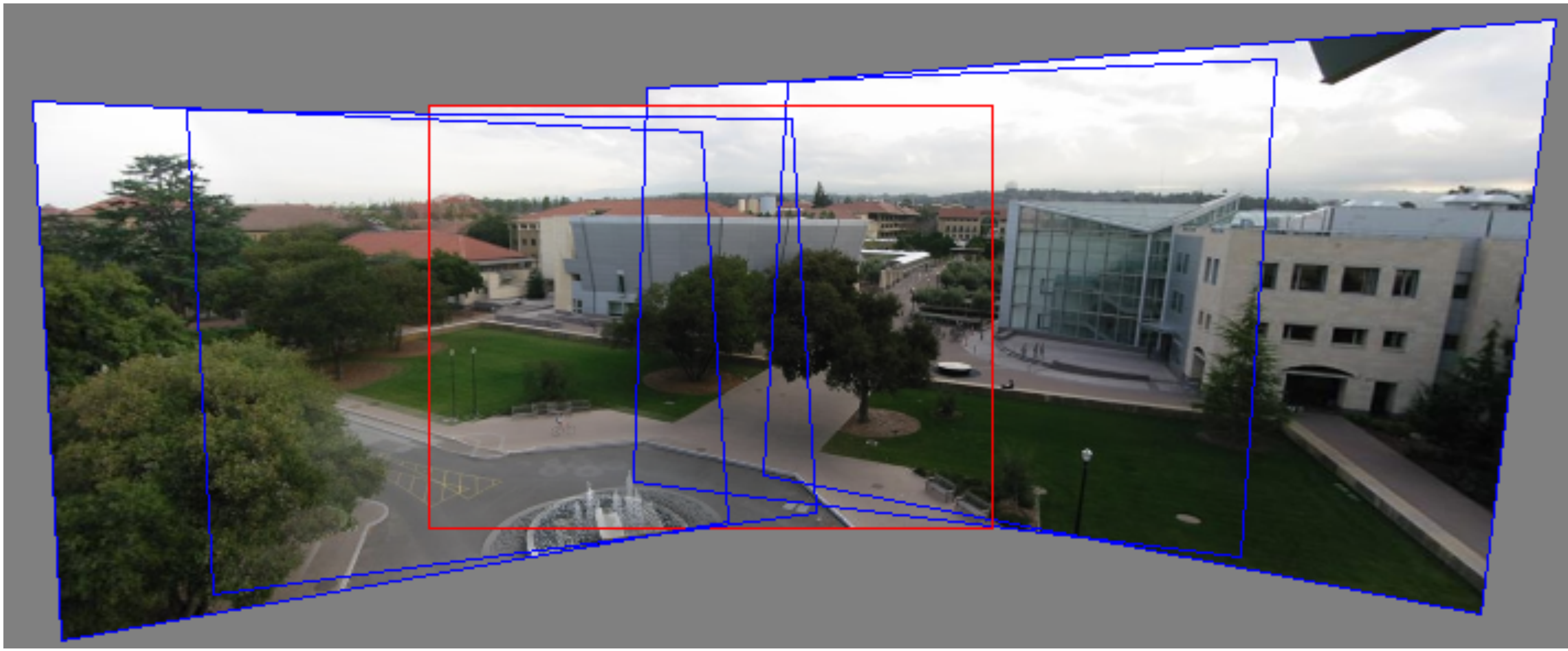
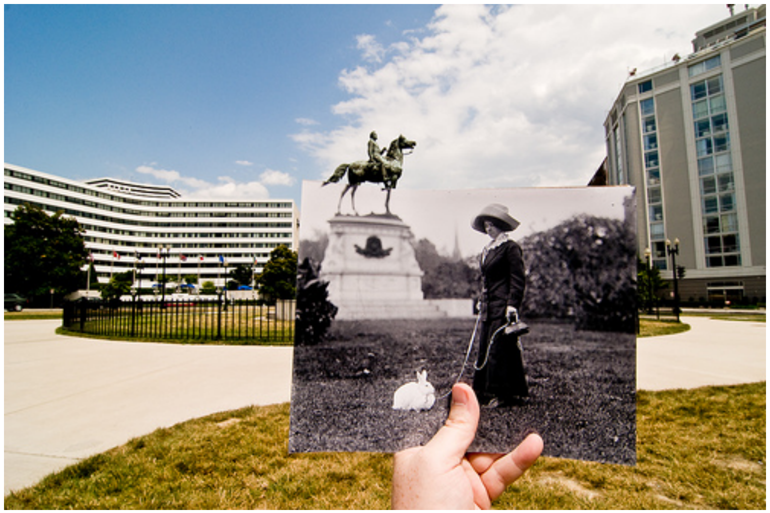


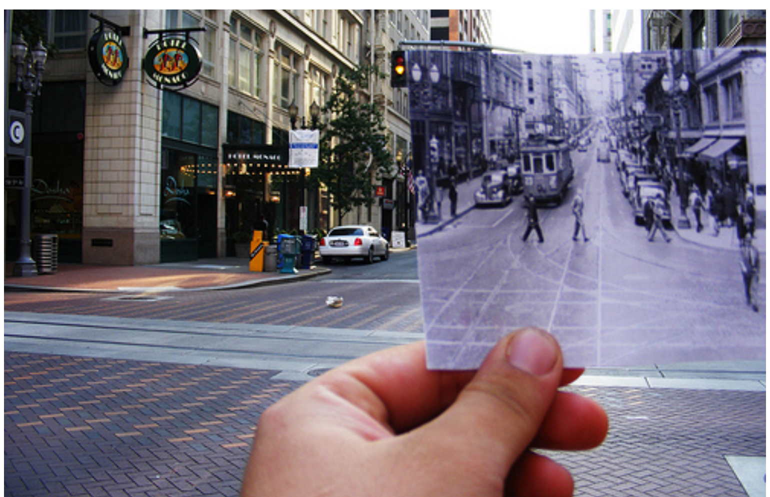
Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fall/

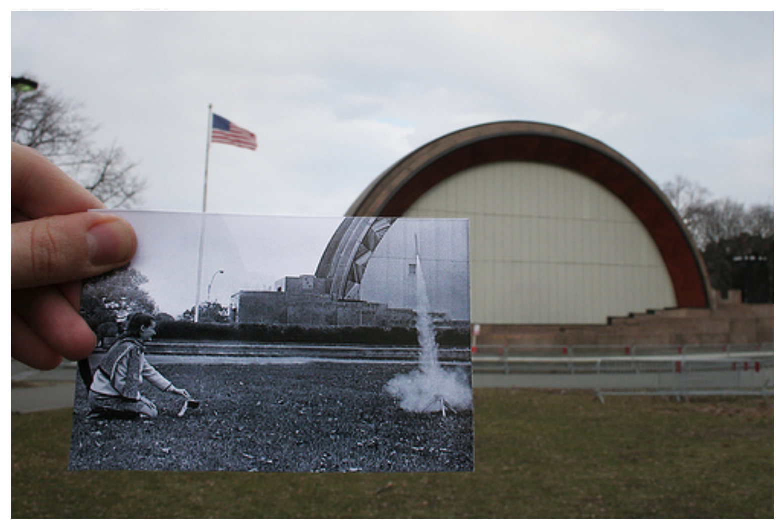
Source: S. Lazebnik

Alignment applications

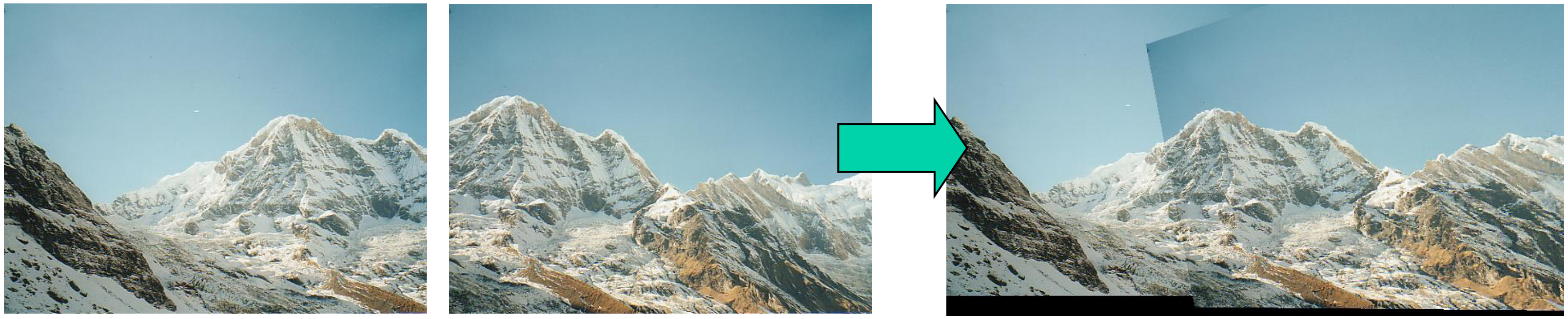
- A look into the past







Alignment applications



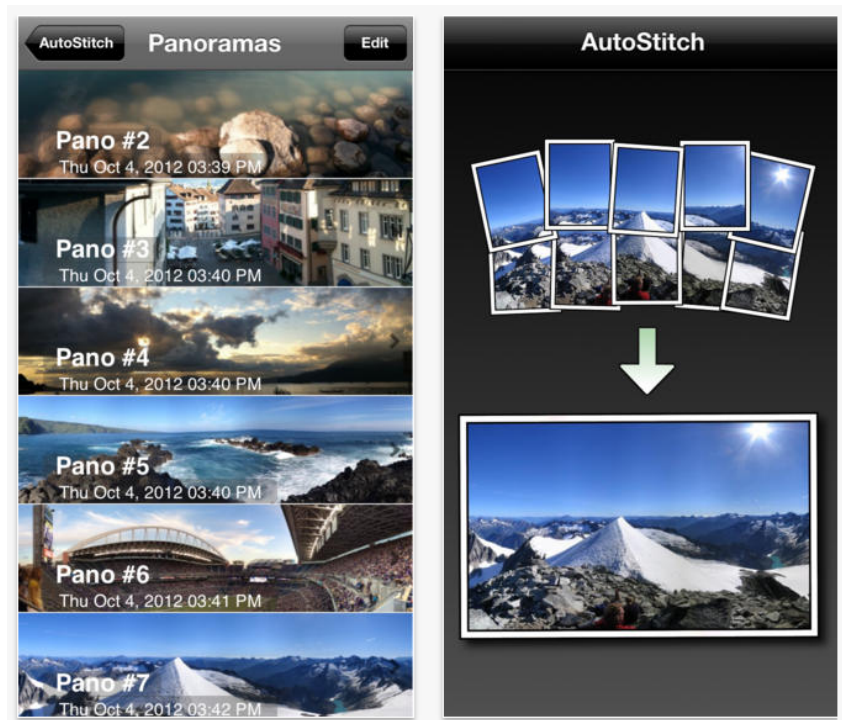
Panorama stitching

AutoStitch Panorama

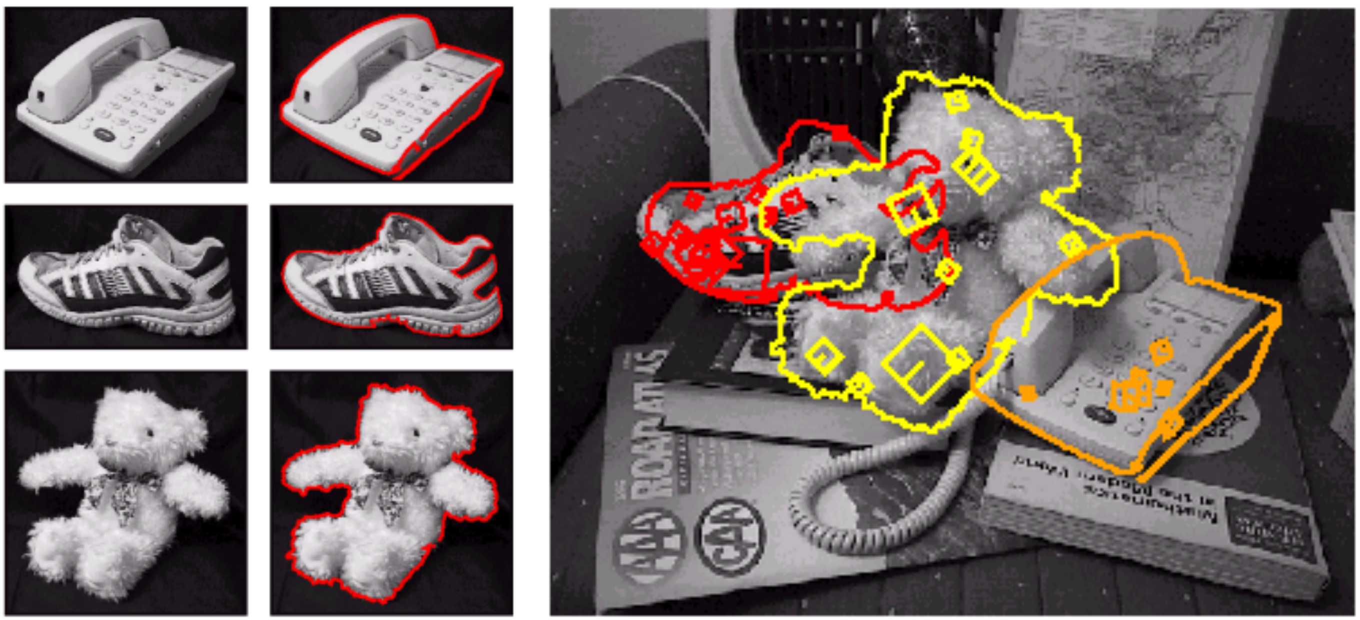
By Cloudburst Research Inc.

Open iTunes to buy and download apps.

View In iTunes



Alignment applications



Recognition of object instances

Source: S. Lazebnik

Alignment

Alignment: find parameters of model that maps one set of points to another

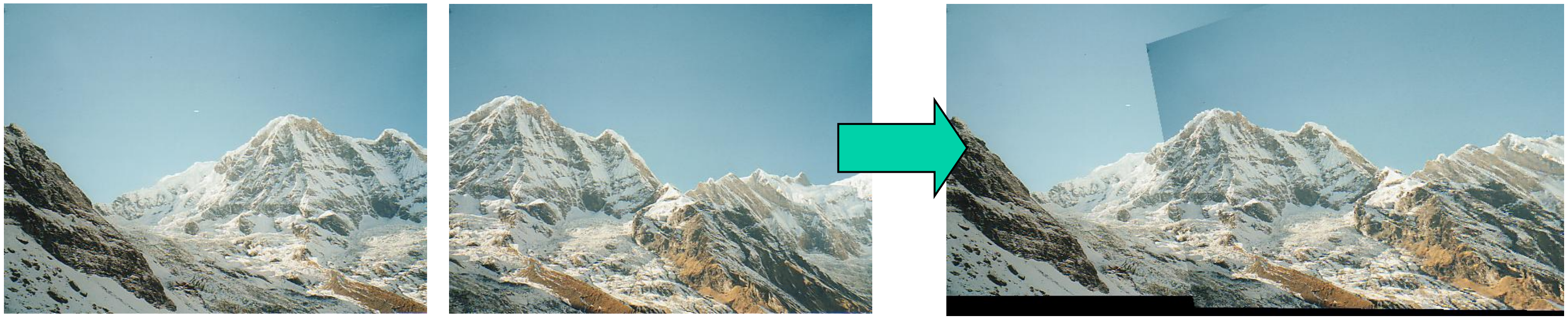
Typically want to solve for a global transformation that accounts for most true correspondences

Difficulties

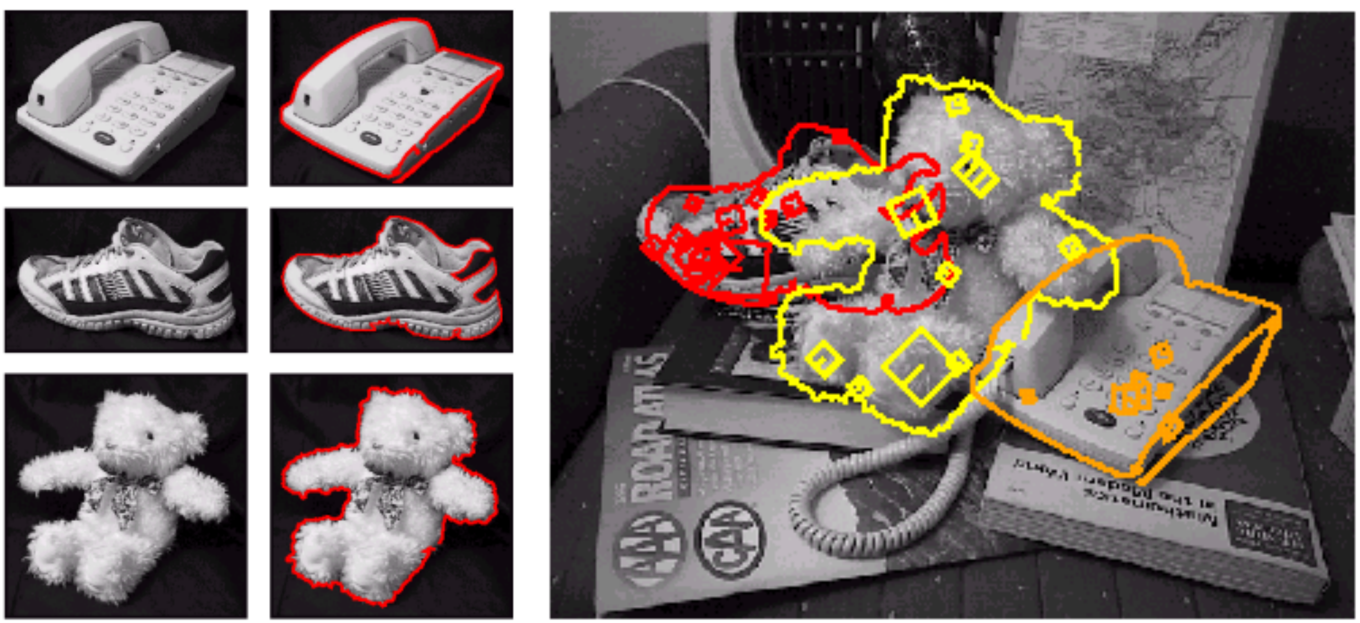
- Noise (typically 1-3 pixels)
- Outliers (often 30-50%)
- Many-to-one matches or multiple objects

Source: S. Lazebnik

Alignment challenges



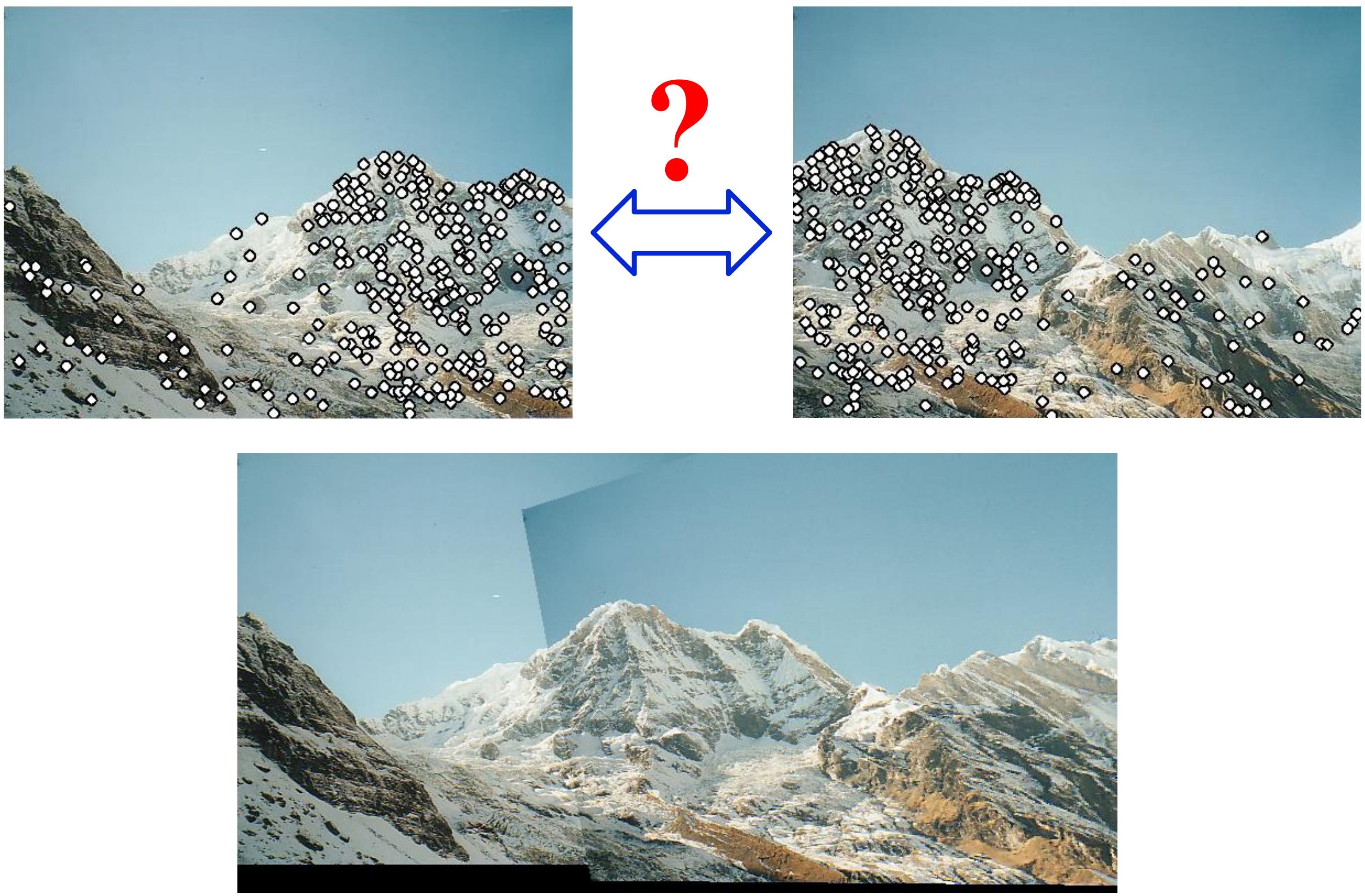
Small degree of overlap
Intensity changes



Occlusion,
clutter

Feature-based alignment

- Search sets of feature matches that agree in terms of:
  a) Local appearance
  b) Geometric configuration



Source: S. Lazebnik

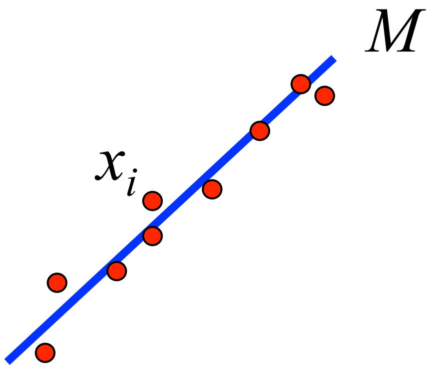
Feature-based alignment: Overview

- Alignment as fitting
  - Affine transformations
  - Homographies

- Robust alignment
  - Descriptor-based feature matching
  - RANSAC

Alignment as fitting

- Previous lectures: fitting a model to features in one image



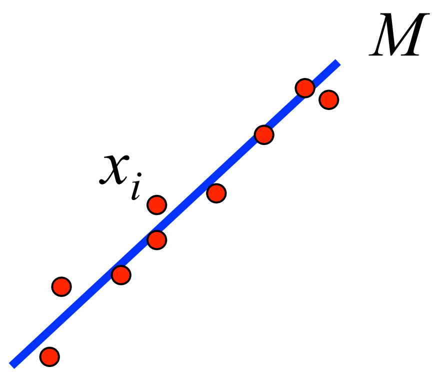
Find model *M* that minimizes

$$\sum_i \text{residual}(x_i, M)$$

Source: S. Lazebnik

Alignment as fitting

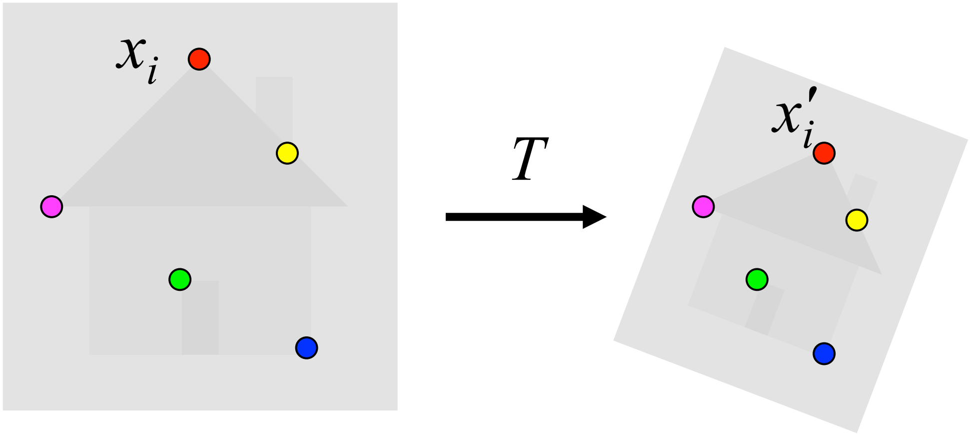
- Previous lectures: fitting a model to features in one image



Find model M that minimizes

$$\sum_i \text{residual}(x_i, M)$$

- Alignment: fitting a model to a transformation between pairs of features (*matches*) in two images



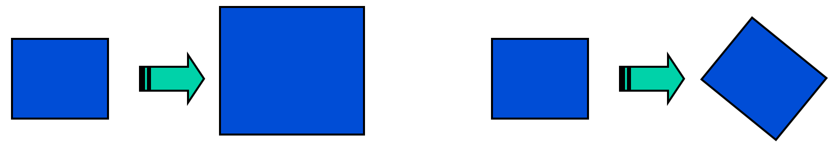
Find transformation T that minimizes

$$\sum_i \text{residual}(T(x_i), x_i')$$

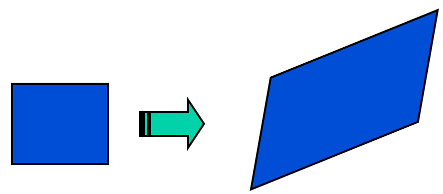
Source: S. Lazebnik

2D transformation models

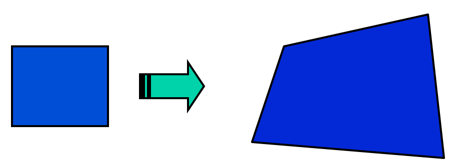
- Similarity (translation, scale, rotation)



- Affine



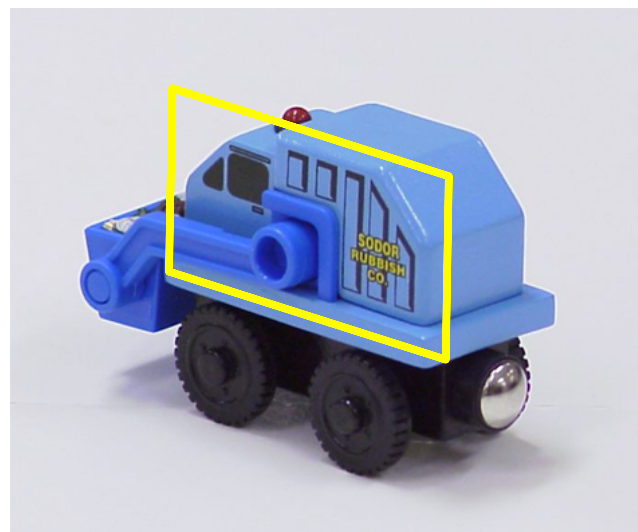
- Projective (homography)



Source: S. Lazebnik

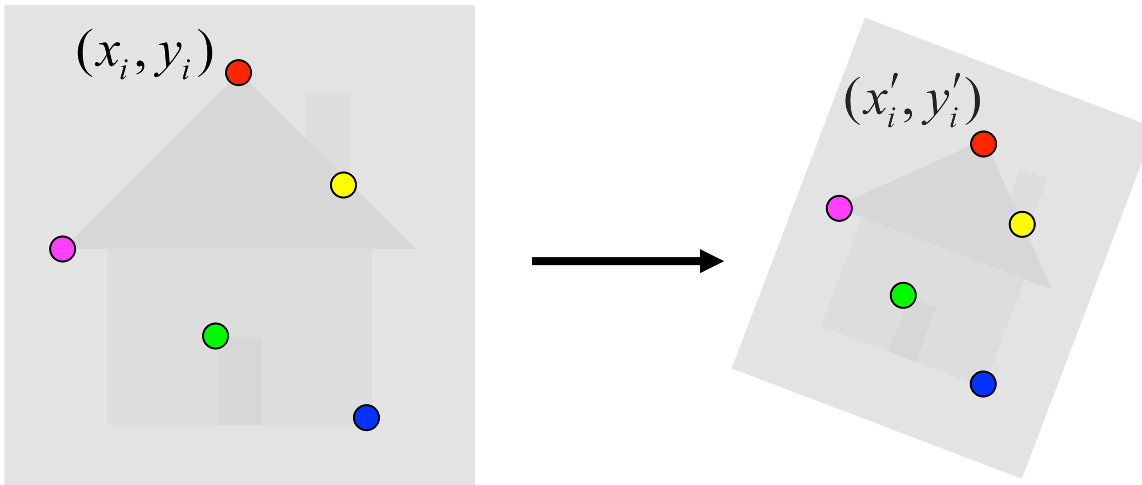
Let's start with affine transformations

- Simple fitting procedure (linear least squares)
- 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?



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

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

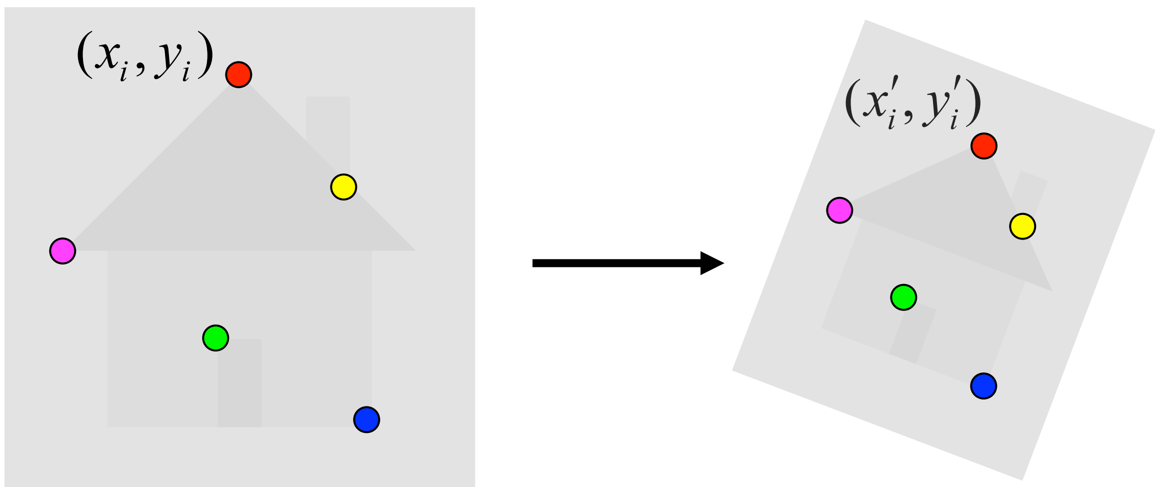
Want to find **M**, **t** to minimize

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

Source: S. Lazebnik

Fitting an affine transformation

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



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

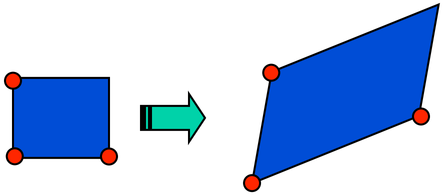
$$\begin{bmatrix} & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Source: S. Lazebnik

Fitting an affine transformation

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

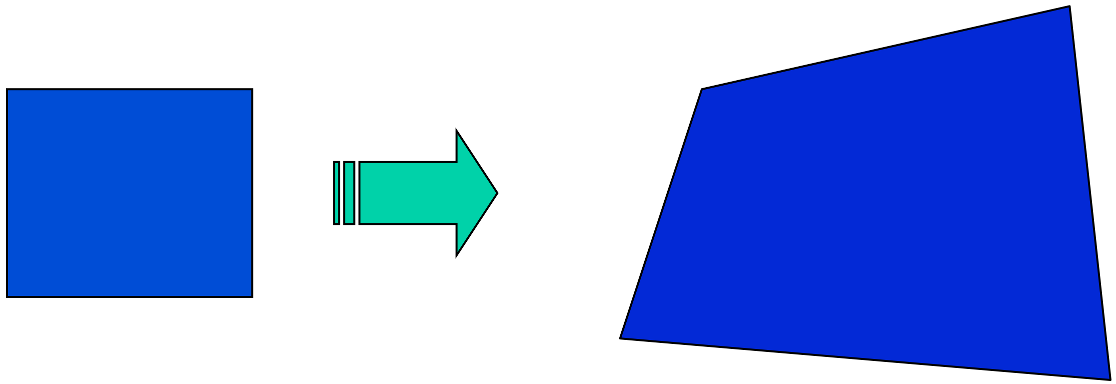
- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters



Source: S. Lazebnik

Fitting a plane projective transformation

- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)

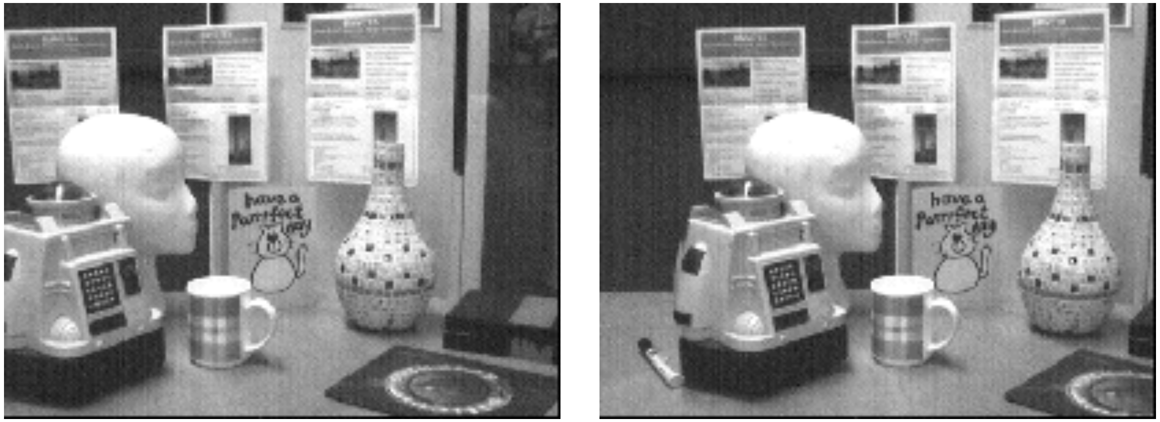


Source: S. Lazebnik

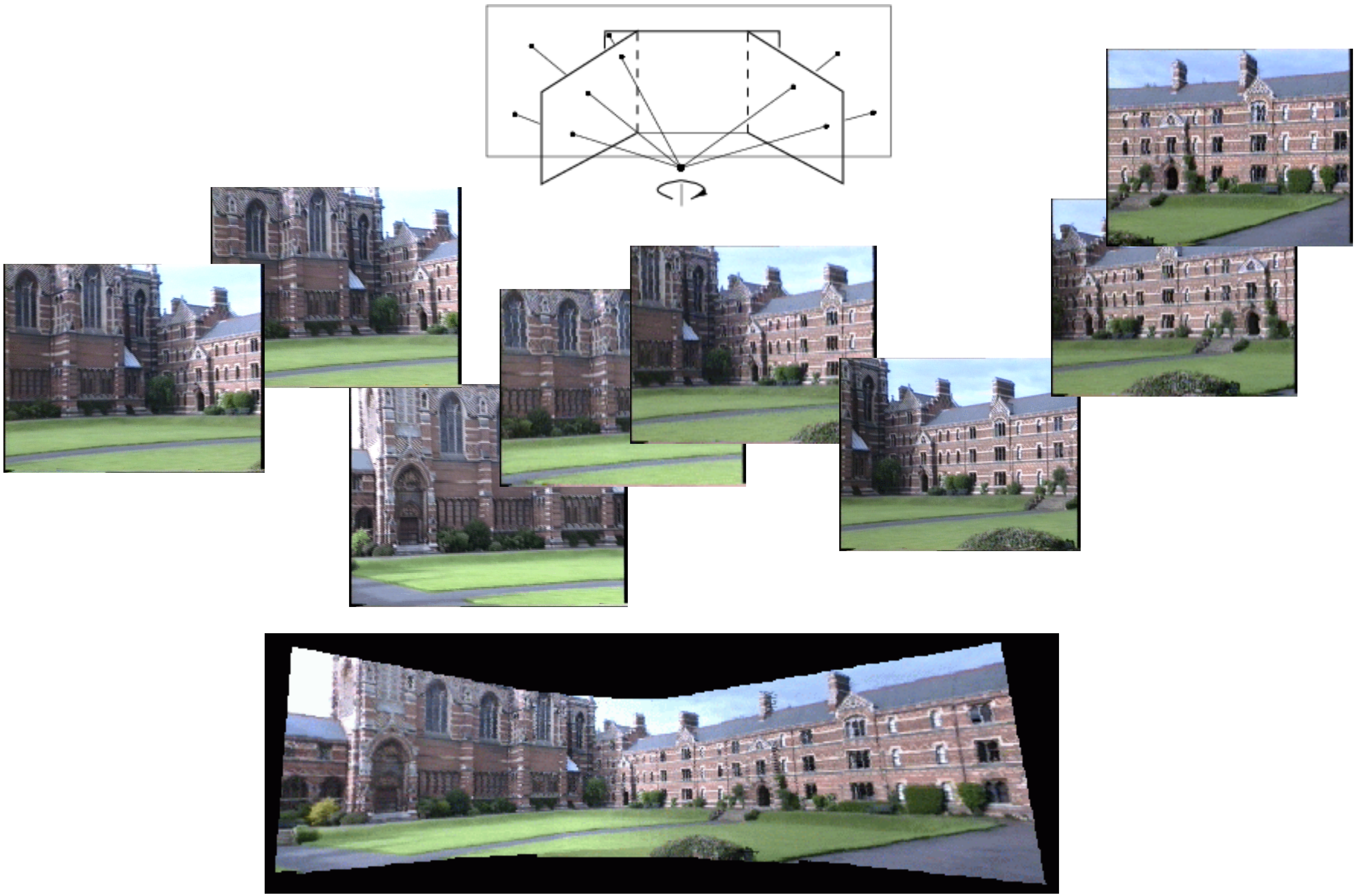
Homography

- The transformation between two views of a planar surface

- The transformation between images from two cameras that share the same center



Application: Panorama stitching



Source: Hartley & Zisserman

Source: S. Lazebnik

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

Source: S. Lazebnik

Fitting a homography

- Recall: homogeneous coordinates

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous image coordinates

- Equation for homography:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Source: S. Lazebnik

Fitting a homography

- Equation for homography:

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\lambda \mathbf{x}_i' = \mathbf{H}\,\mathbf{x}_i$$

$$\mathbf{x}_i' \times \mathbf{H}\,\mathbf{x}_i = 0$$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h}_1^T \mathbf{x}_i \\ \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_3^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x_i' \mathbf{h}_3^T \mathbf{x}_i \\ x_i' \mathbf{h}_2^T \mathbf{x}_i - y_i' \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y_i' \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x_i' \mathbf{x}_i^T \\ -y_i' \mathbf{x}_i^T & x_i' \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

3 equations, only 2 linearly independent

Source: S. Lazebnik

Fitting a homography: DLT algorithm

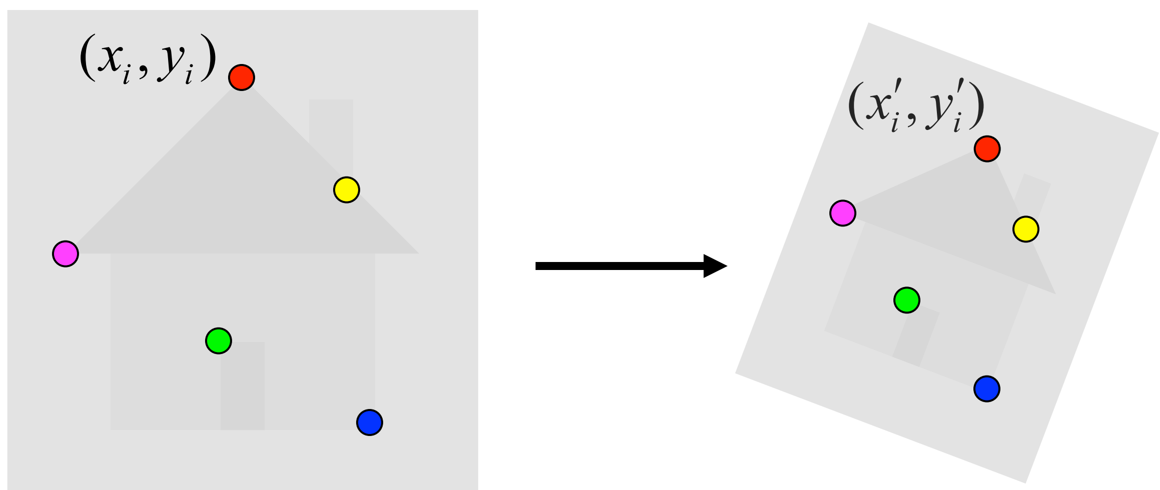
$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y_1'\,\mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x_1'\,\mathbf{x}_1^T \\ \cdots & \cdots & \cdots \\ 0^T & \mathbf{x}_n^T & -y_n'\,\mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x_n'\,\mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \qquad \mathbf{A}\,\mathbf{h} = 0$$

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Homogeneous least squares: find $\mathbf{h}$ minimizing $\|\mathbf{A}\mathbf{h}\|^2$
  - Eigenvector of $\mathbf{A}^{\mathbf{T}}\mathbf{A}$ corresponding to smallest eigenvalue
  - Four matches needed for a minimal solution
- For more info, see Sec. 4.1 in (Hartley & Zisserman)

Source: S. Lazebnik

Robust feature-based alignment

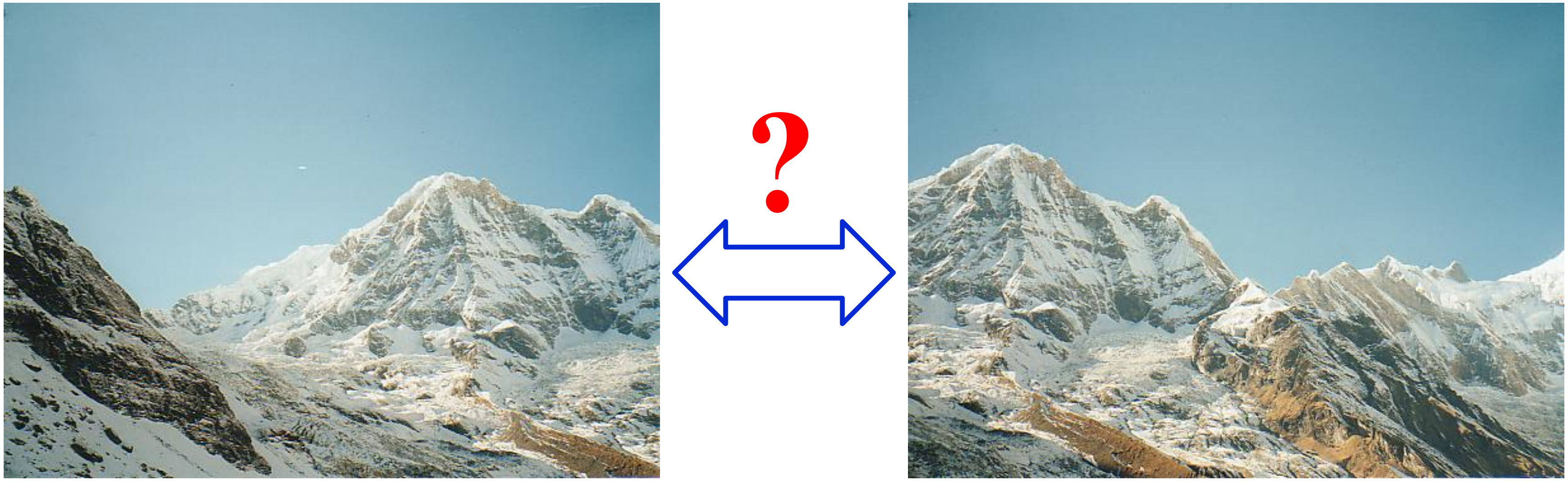
- So far, we've assumed that we are given a set of "ground-truth" correspondences between the two images we want to align
- What if we don't know the correspondences?



Source: S. Lazebnik

Robust feature-based alignment

- So far, we've assumed that we are given a set of "ground-truth" correspondences between the two images we want to align
- What if we don't know the correspondences?

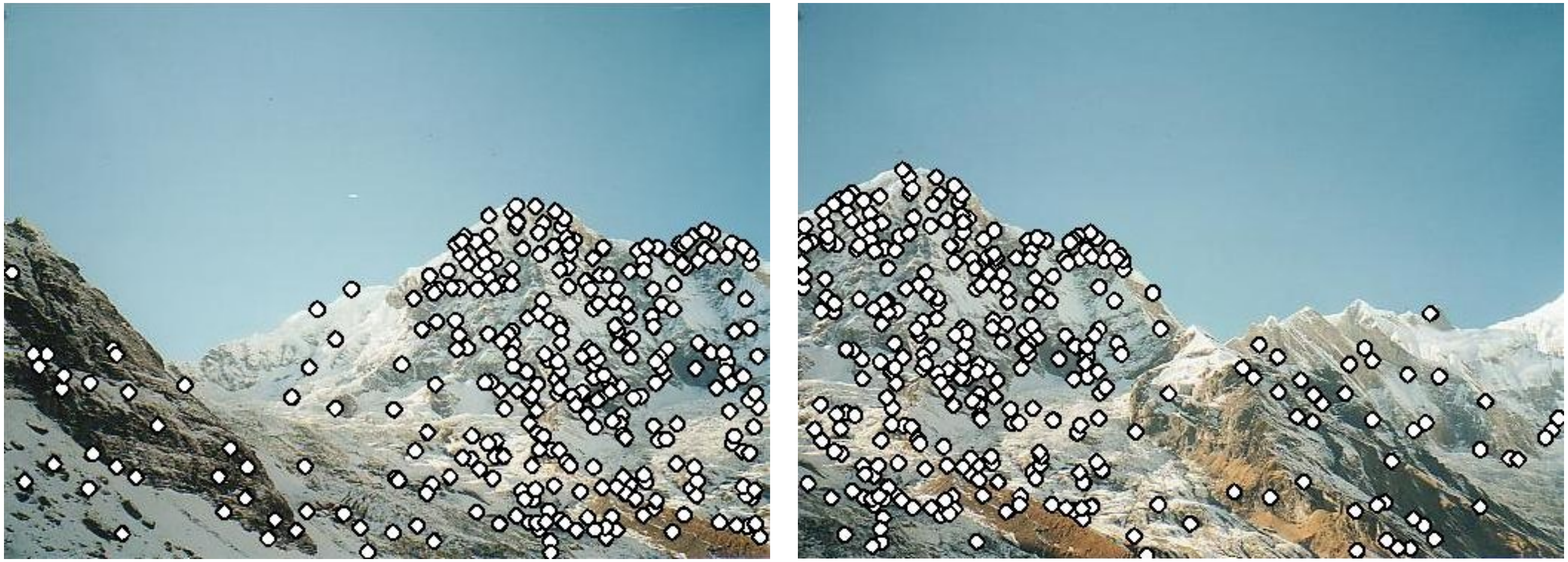


Source: S. Lazebnik

Robust feature-based alignment

Source: S. Lazebnik

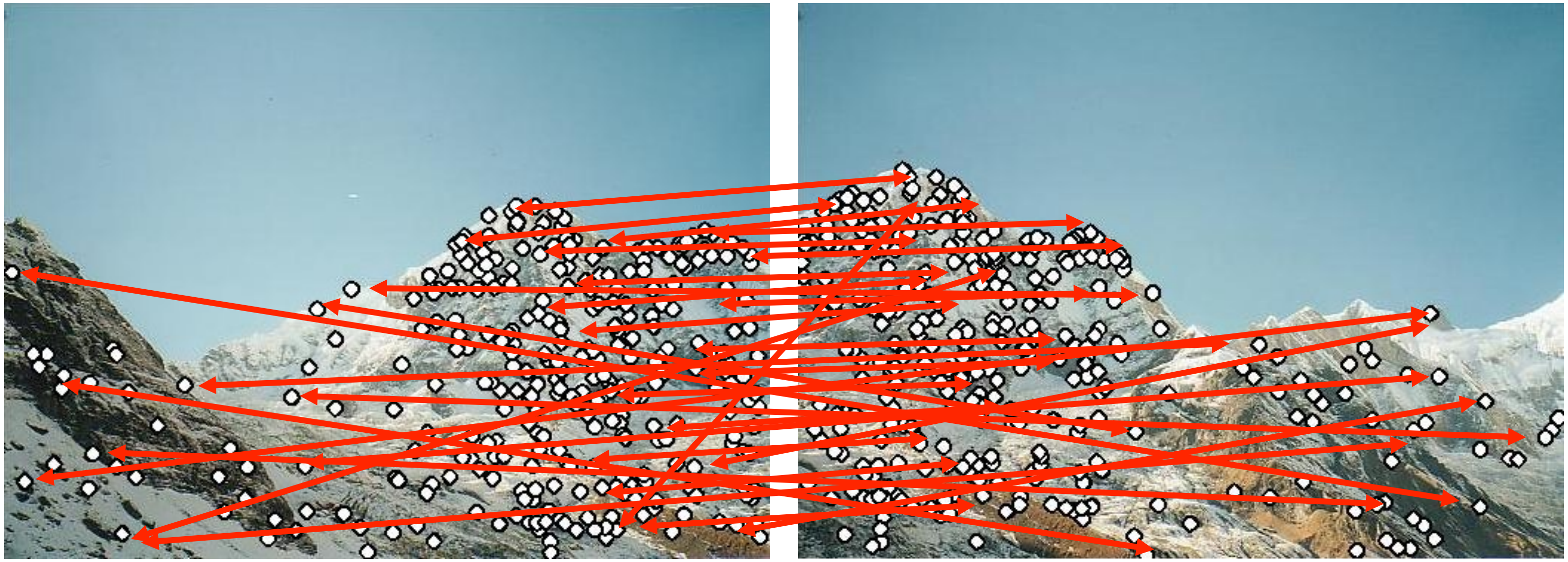
Robust feature-based alignment



- Extract features

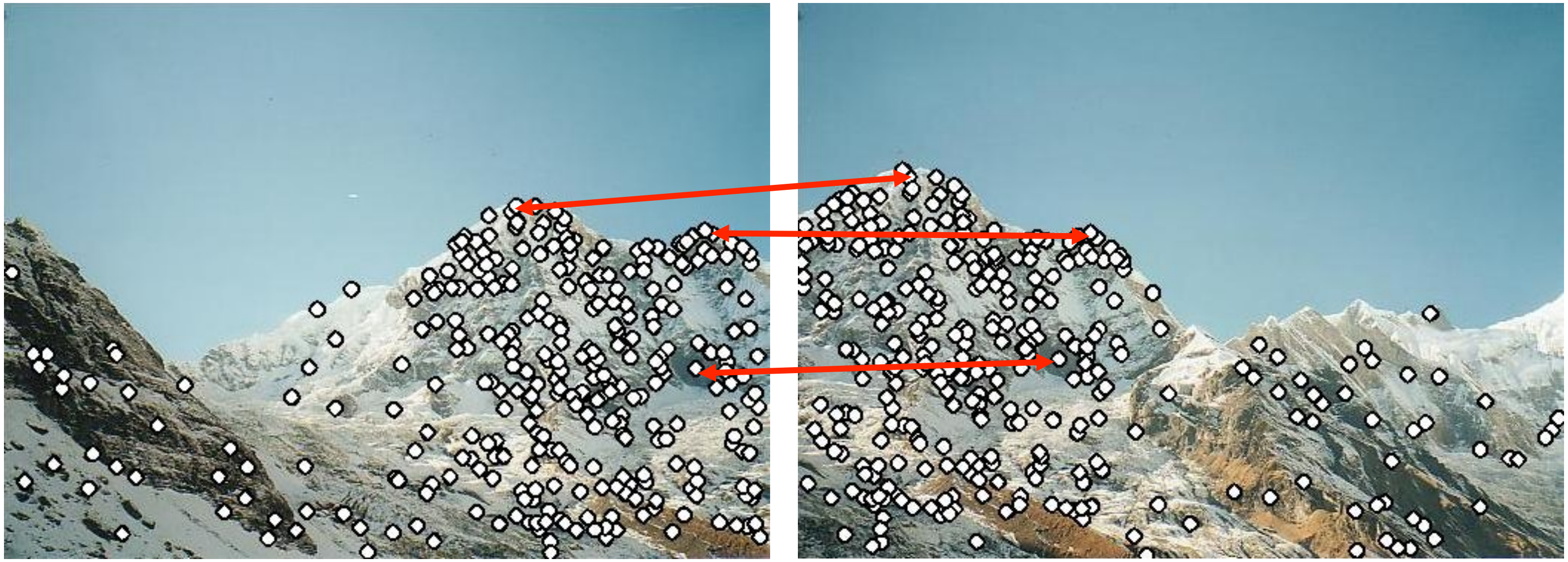
Source: S. Lazebnik

Robust feature-based alignment



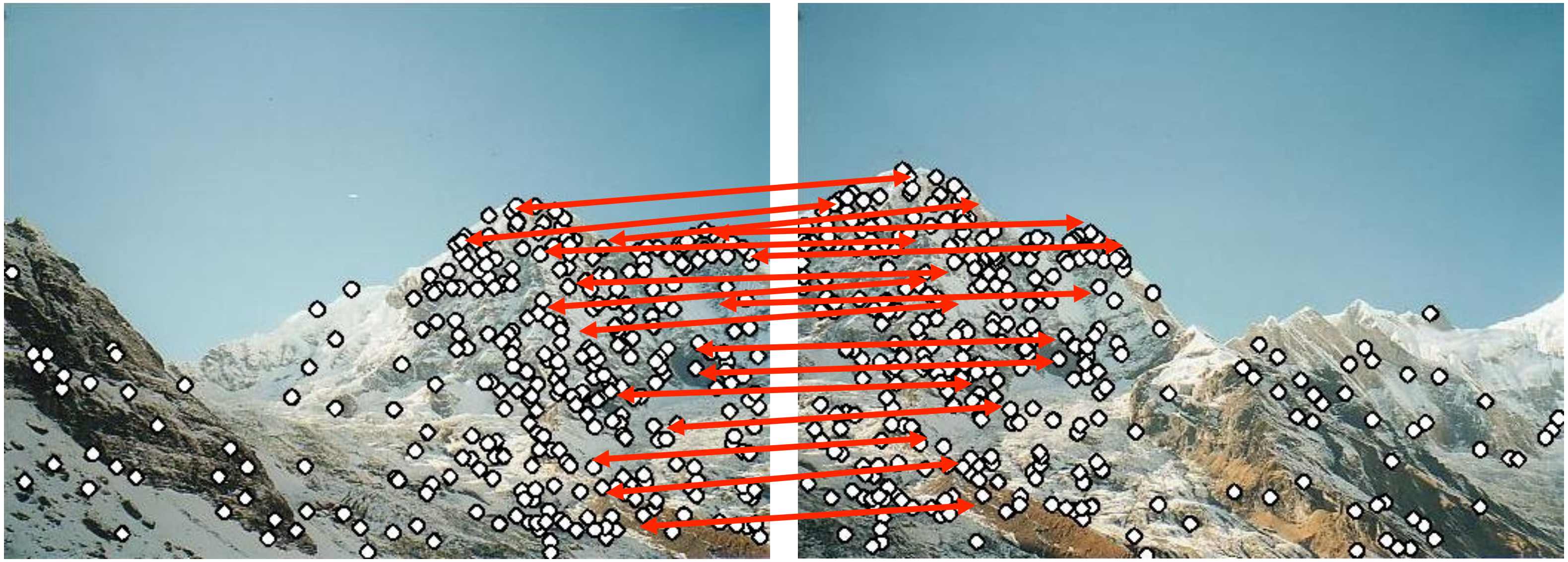
- Extract features
- Compute *putative matches*

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation T

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
    - *Hypothesize* transformation *T*
    - *Verify* transformation (search for other matches consistent with *T*)

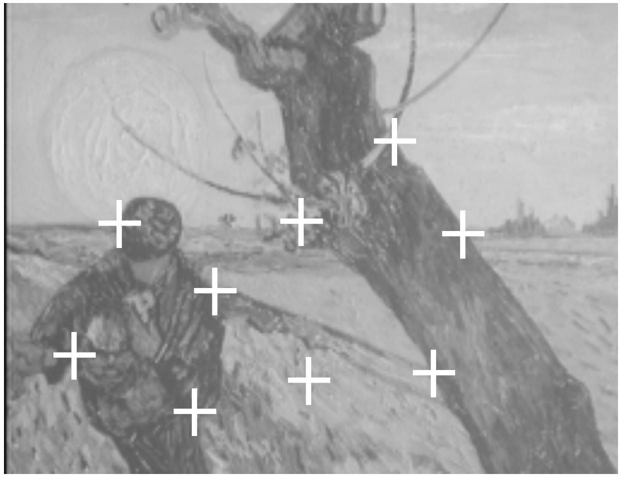
Source: S. Lazebnik

Robust feature-based alignment

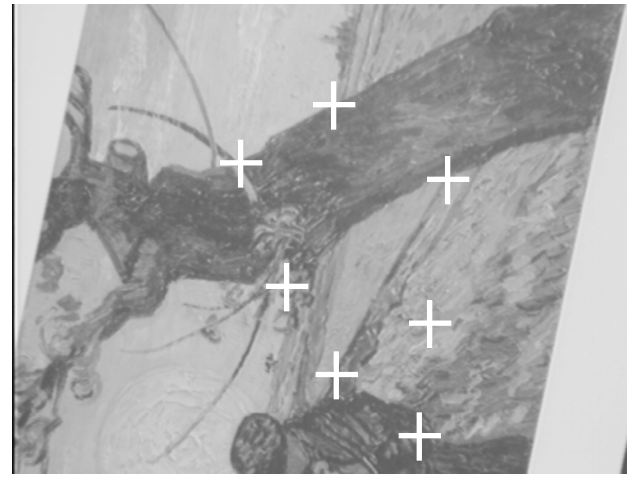
- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation T
  - *Verify* transformation (search for other matches consistent with T)

Source: S. Lazebnik

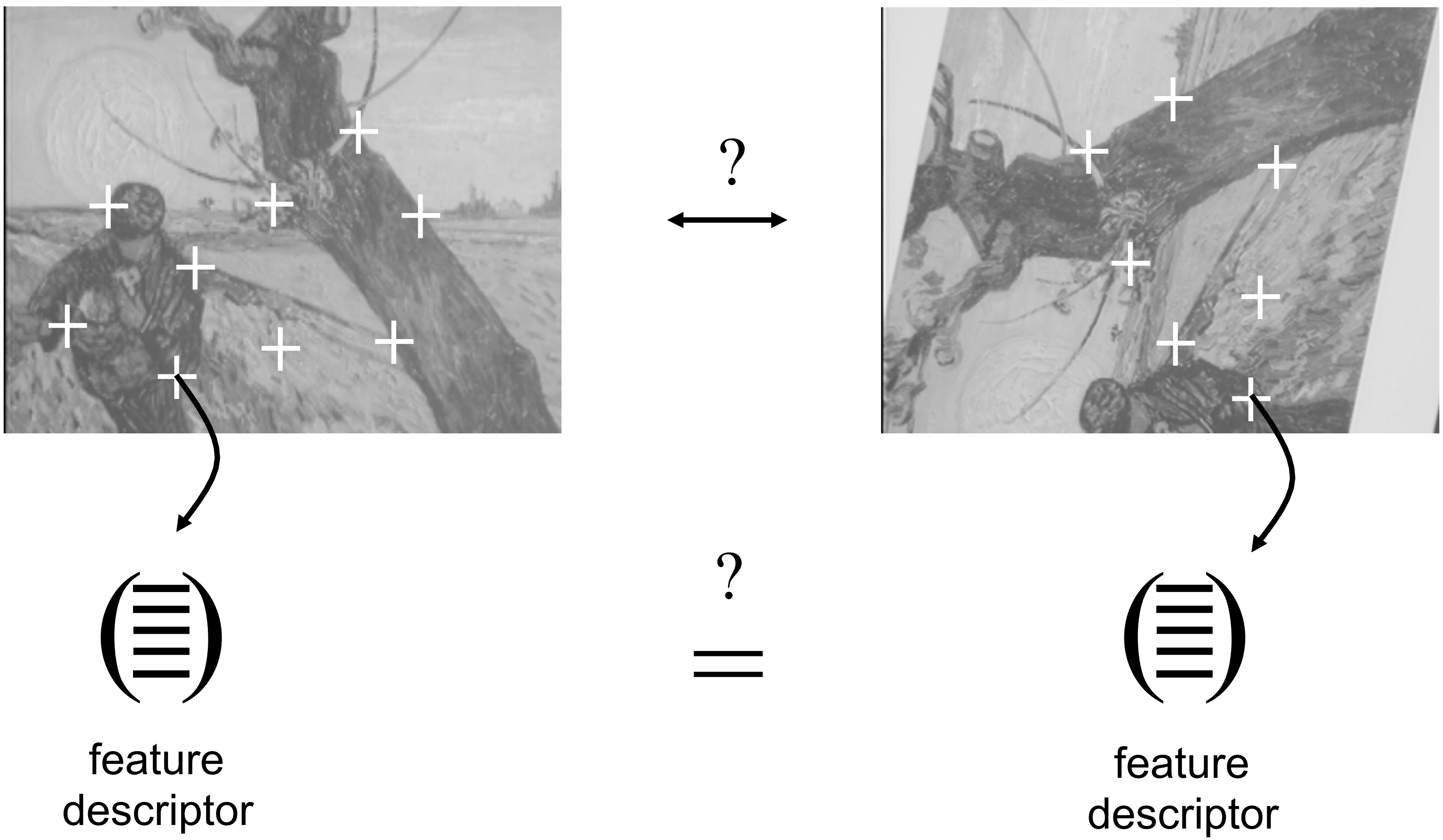
Generating putative correspondences



?



Generating putative correspondences

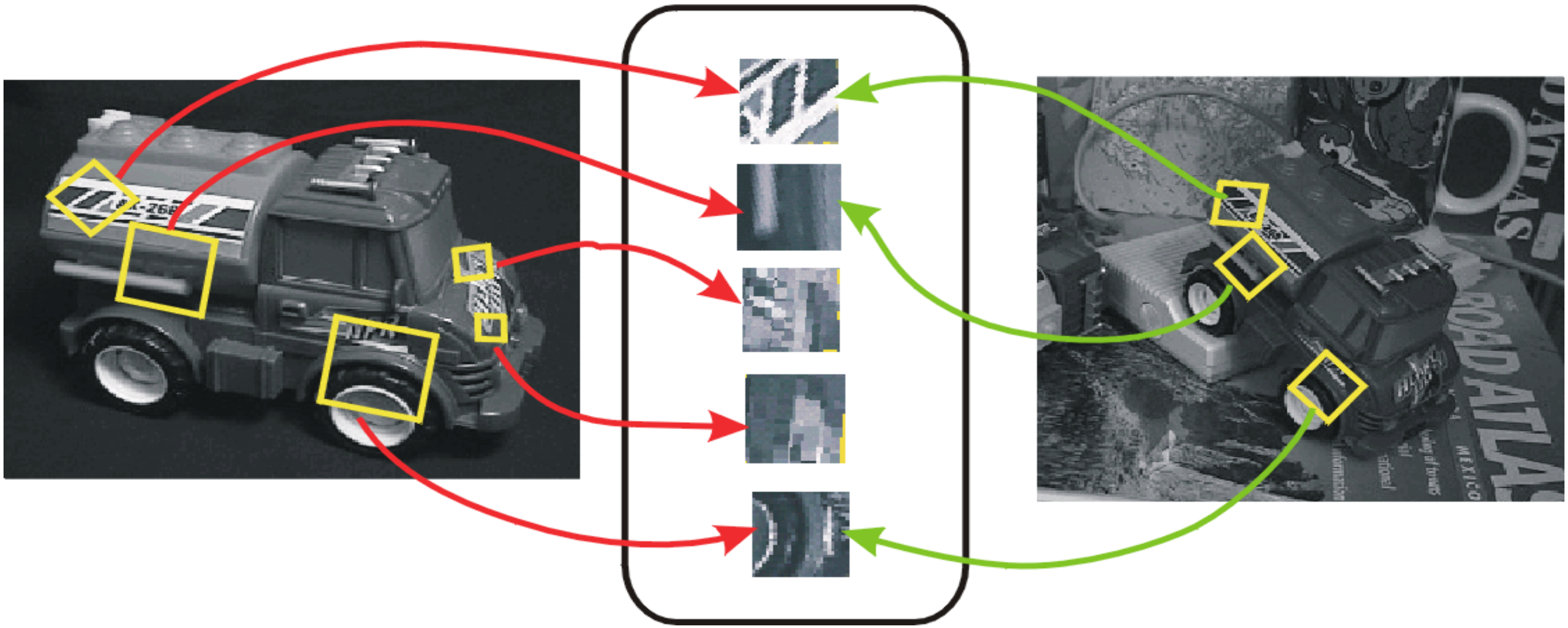


- Need to compare *feature descriptors* of local patches surrounding interest points

Source: S. Lazebnik

Feature descriptors

- Recall: feature detection and description



Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
  - Sum of squared differences (SSD)

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

    – Not invariant to intensity change

  - Normalized correlation

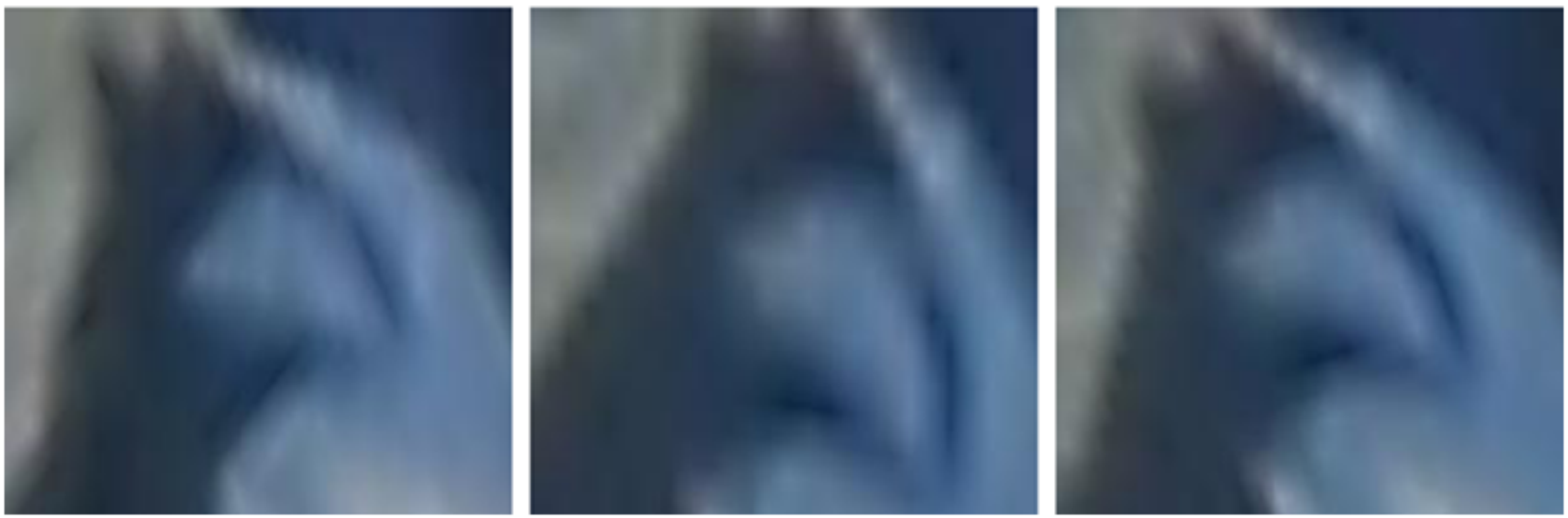
$$\rho(\mathbf{u},\mathbf{v}) = \frac{(\mathbf{u}-\overline{\mathbf{u}})}{\|\mathbf{u}-\overline{\mathbf{u}}\|} \cdot \frac{(\mathbf{v}-\overline{\mathbf{v}})}{\|\mathbf{v}-\overline{\mathbf{v}}\|} = \frac{\sum_i (u_i - \overline{\mathbf{u}})(v_i - \overline{\mathbf{v}})}{\sqrt{\left(\sum_j (u_j - \overline{\mathbf{u}})^2\right)\left(\sum_j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

    – Invariant to affine intensity change

Source: S. Lazebnik

Disadvantage of intensity vectors as descriptors

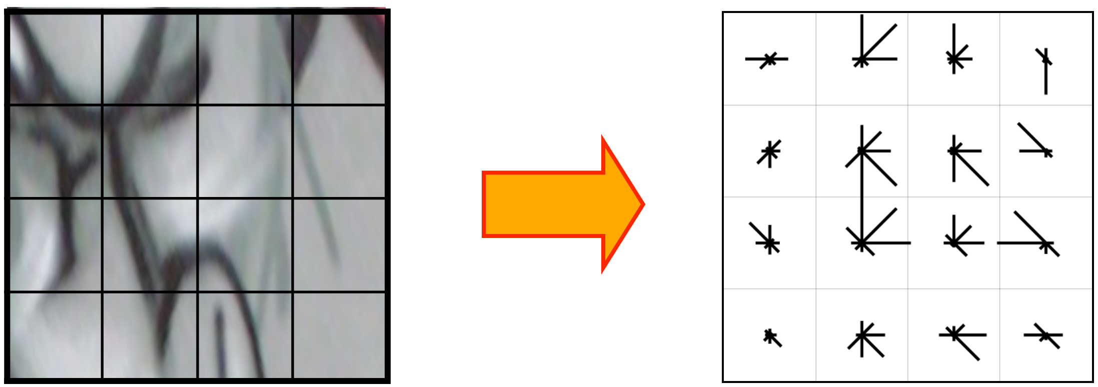
- Small deformations can affect the matching score a lot



Source: S. Lazebnik

Feature descriptors: SIFT

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



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

Source: S. Lazebnik

Feature descriptors: SIFT

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

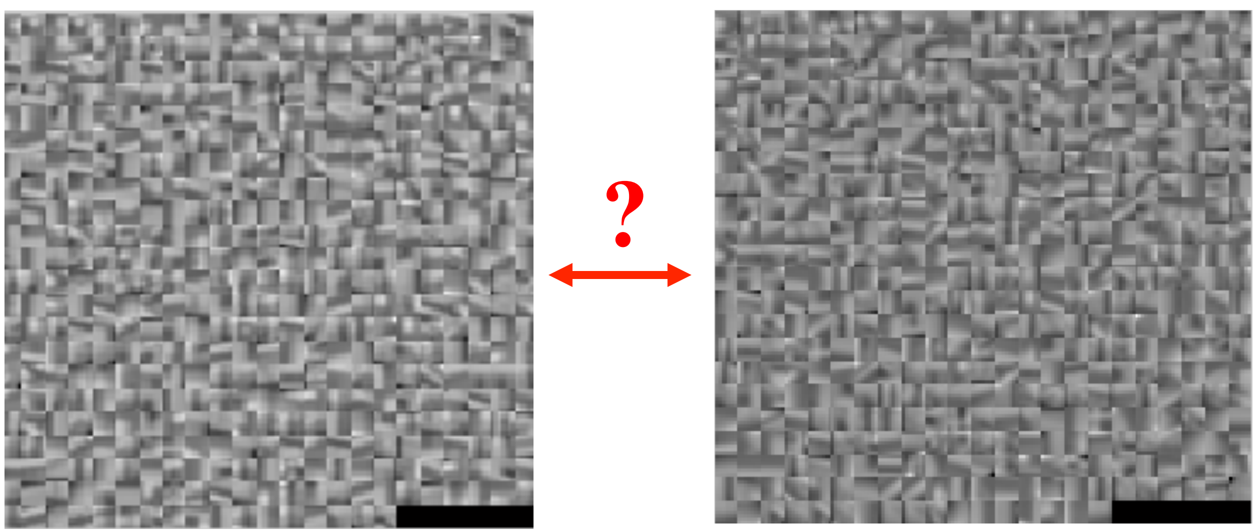
- Advantage over raw vectors of pixel values
  - Gradients less sensitive to illumination change
  - Pooling of gradients over the sub-patches achieves robustness to small shifts, but still preserves some spatial information

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

Source: S. Lazebnik

Feature matching

- Generating *putative matches*: for each patch in one image, find a short list of patches in the other image that could match it based solely on appearance



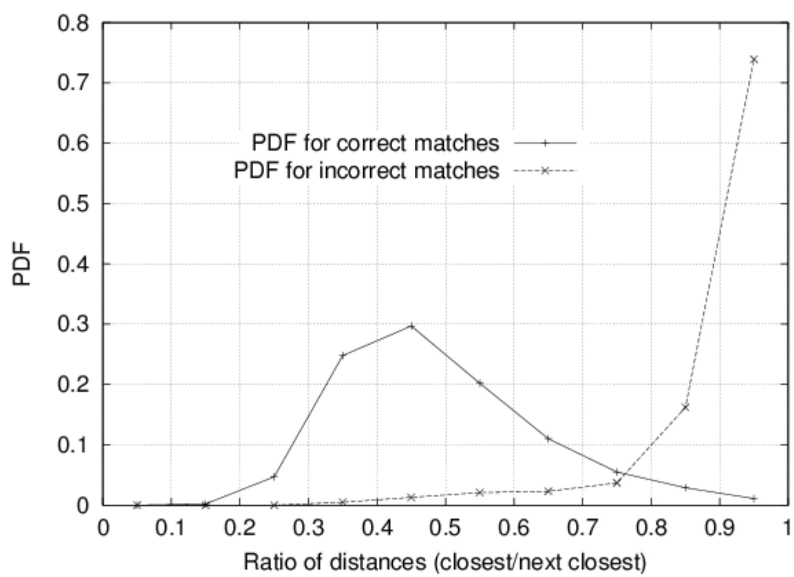
Source: S. Lazebnik

Problem: Ambiguous putative matches

Source: Y. Furukawa

Rejection of unreliable matches

- How can we tell which putative matches are more reliable?
- Heuristic: compare distance of **nearest** neighbor to that of **second** nearest neighbor
  - Ratio of closest distance 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.

Source: S. Lazebnik

RANSAC

- The set of putative matches contains a very high percentage of outliers

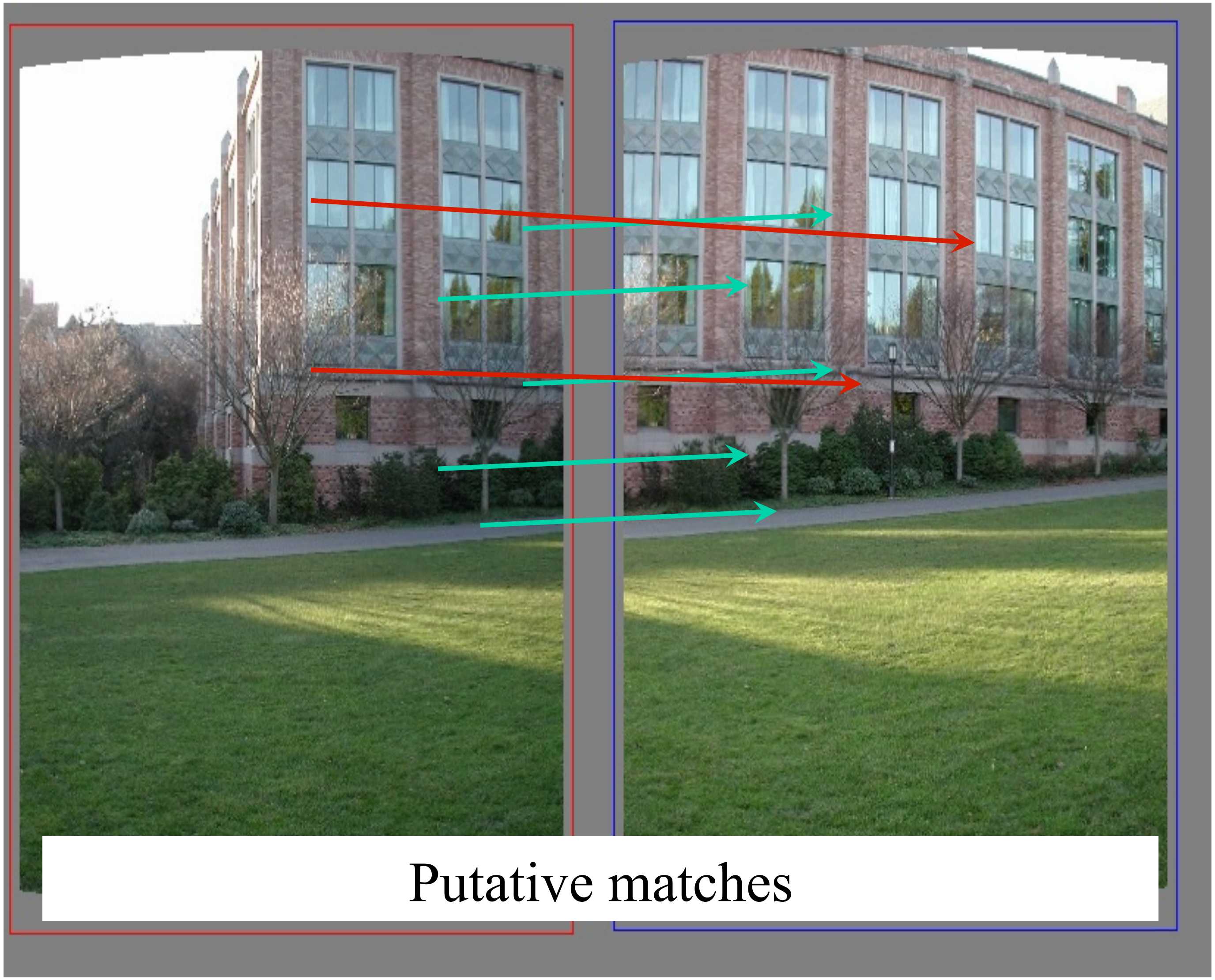
RANSAC loop:

1. Randomly select a *seed 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 least-squares estimate of transformation on all of the inliers

Keep the transformation with the largest number of inliers

Source: S. Lazebnik

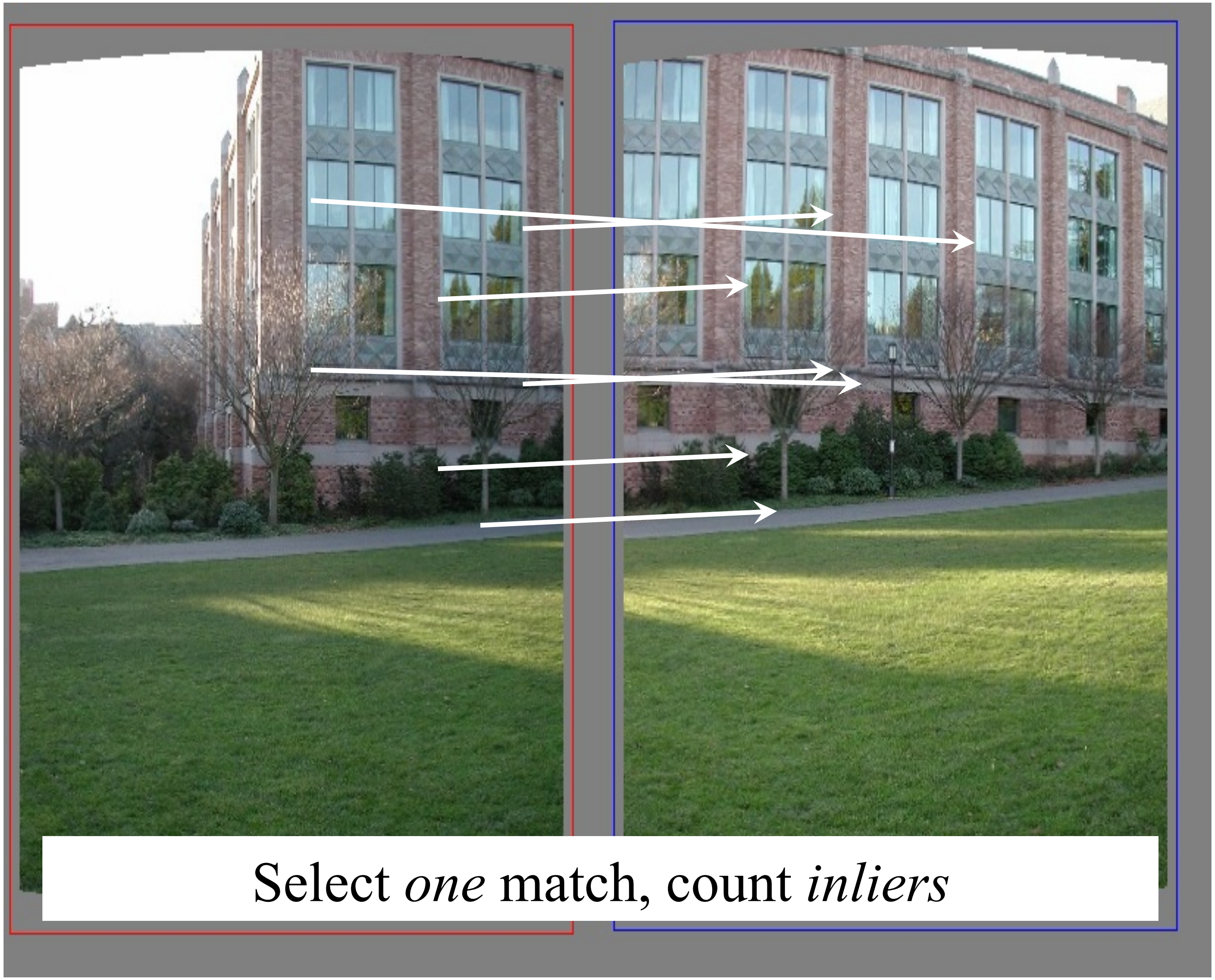
RANSAC example: Translation



Putative matches

Source: S. Lazebnik

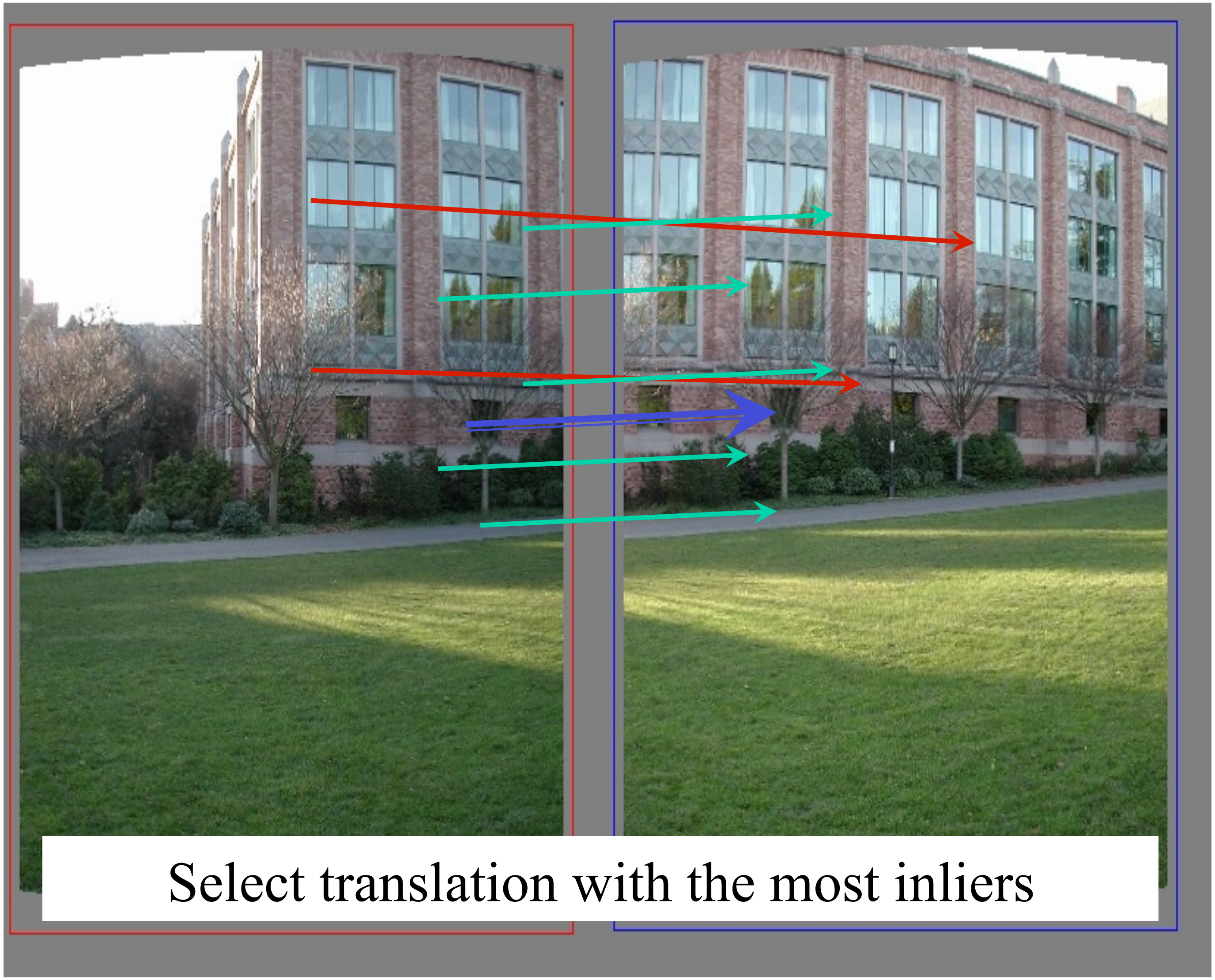
RANSAC example: Translation



Select *one* match, count *inliers*

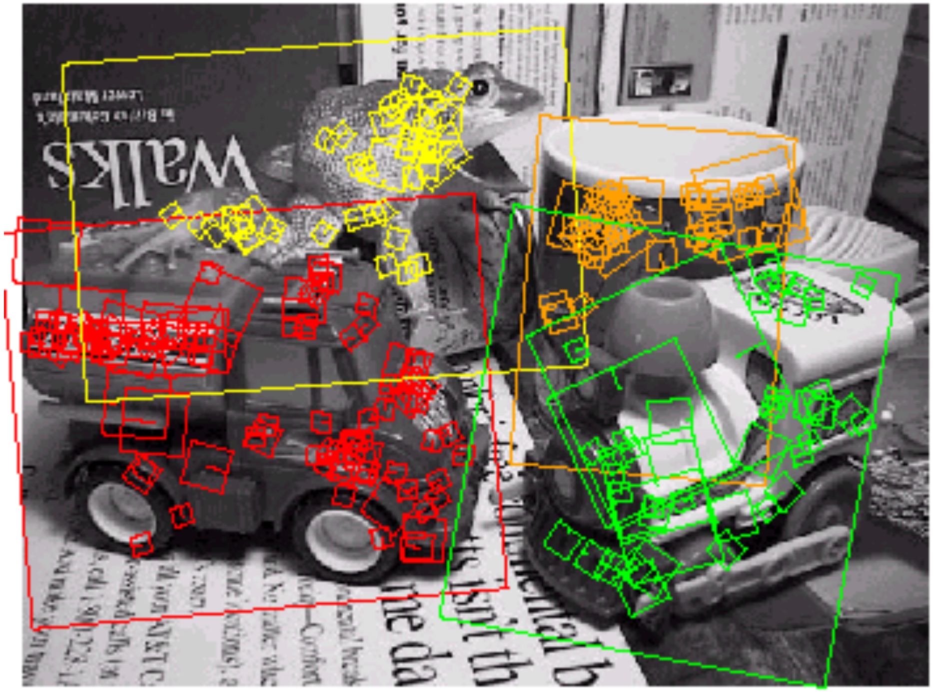
Source: S. Lazebnik

RANSAC example: Translation



Select translation with the most inliers

Object Instance Recognition



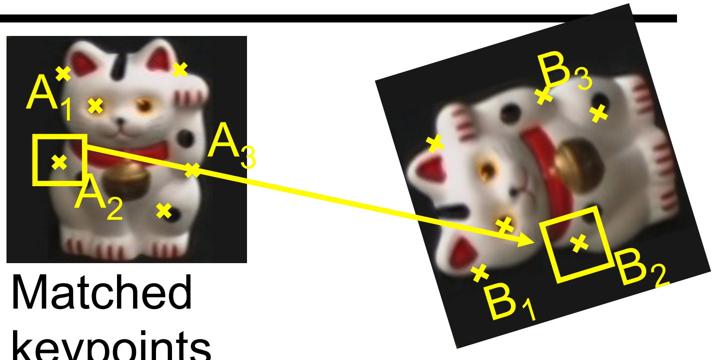
Slide credit: David Lowe

Object Instance Recognition

1. Match keypoints to object model

2. Solve for affine transformation parameters

3. Score by inliers and choose solutions with score above threshold



Matched keypoints

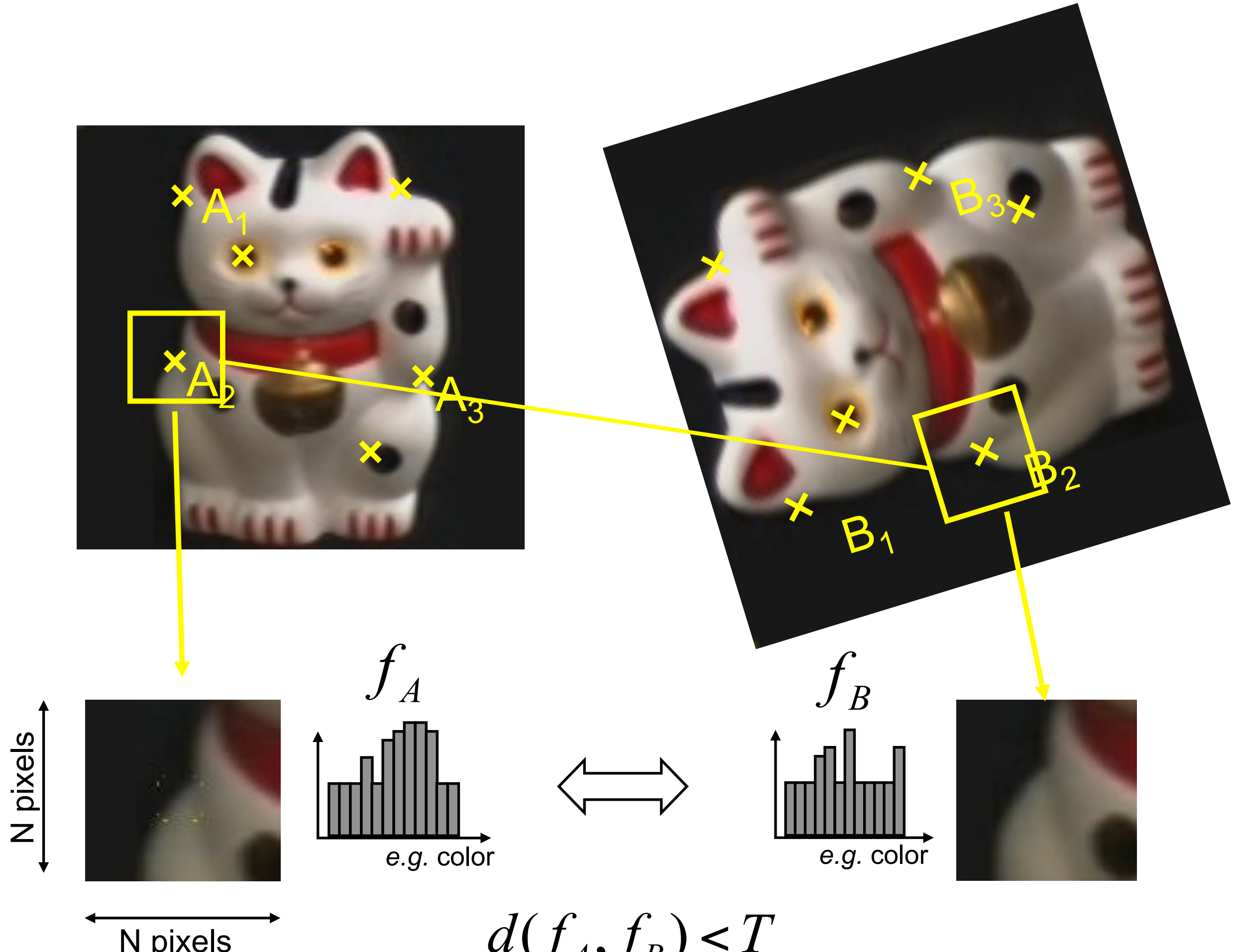
Affine Parameters

Inliers

Choose hypothesis with max score above threshold

Source: D. Hoeim

Overview of Keypoint Matching



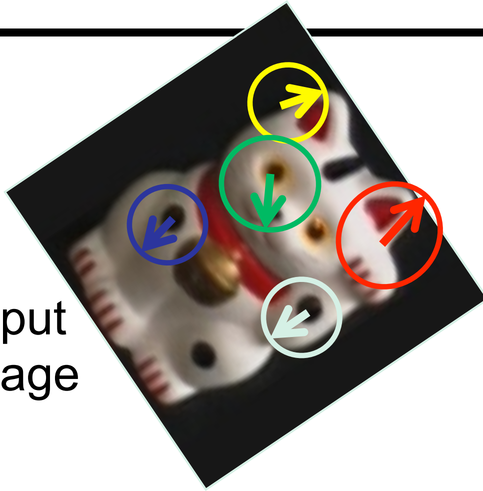
$$d(f_A, f_B) < T$$

K. Grauman, B. Leibe

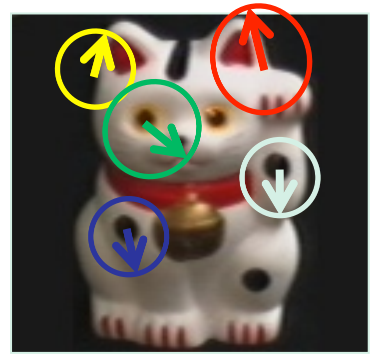
1. **Find a set of distinctive key-points**
2. **Define a region around each keypoint**
3. **Extract and normalize the region content**
4. **Compute a local descriptor from the normalized region**
5. **Match local descriptors**

Finding the objects (overview)

Source: D. Hoeim



Input Image



Stored Image

1. Match interest points from input image to database image
2. Matched points vote for rough position/orientation/scale of object
3. Find position/orientation/scales that have at least three votes
4. Compute affine registration and matches using iterative least squares with outlier check
5. Report object if there are at least T matched points

Source: D. Hoeim

Matching Keypoints

Want to match keypoints between:

1. Query image
2. Stored image containing the object

Given descriptor x_0, find two nearest neighbors x_1, x_2 with distances d_1, d_2

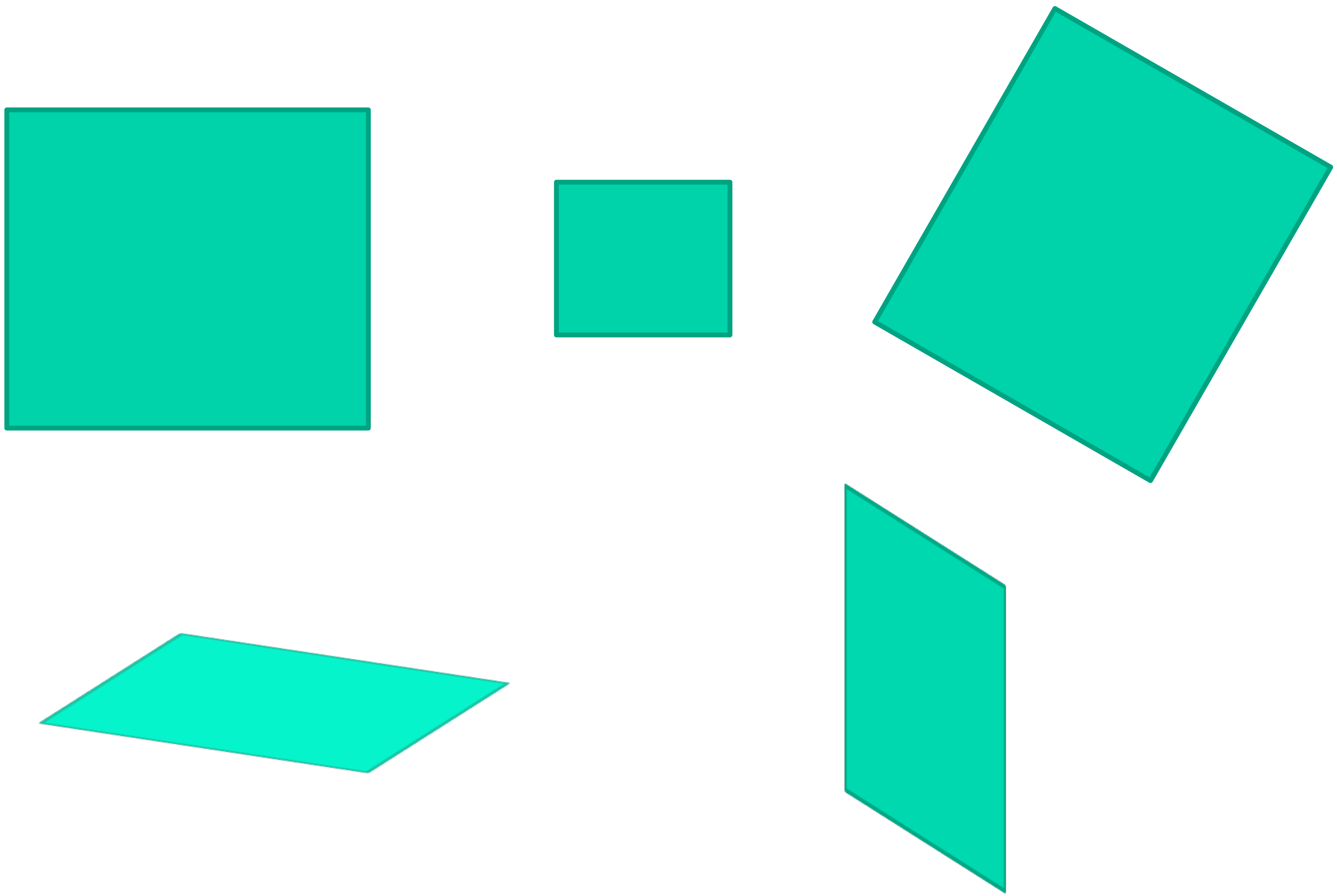
x_1 matches x_0 if $d_1/d_2 < 0.8$

- This gets rid of 90% false matches, 5% of true matches in Lowe's study

Source: D. Hoeim

Affine Object Model

Accounts for 3D rotation of a surface under orthographic projection



Source: D. Hoeim

Affine Object Model

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ & & \vdots & & & \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ \vdots \end{bmatrix}$$

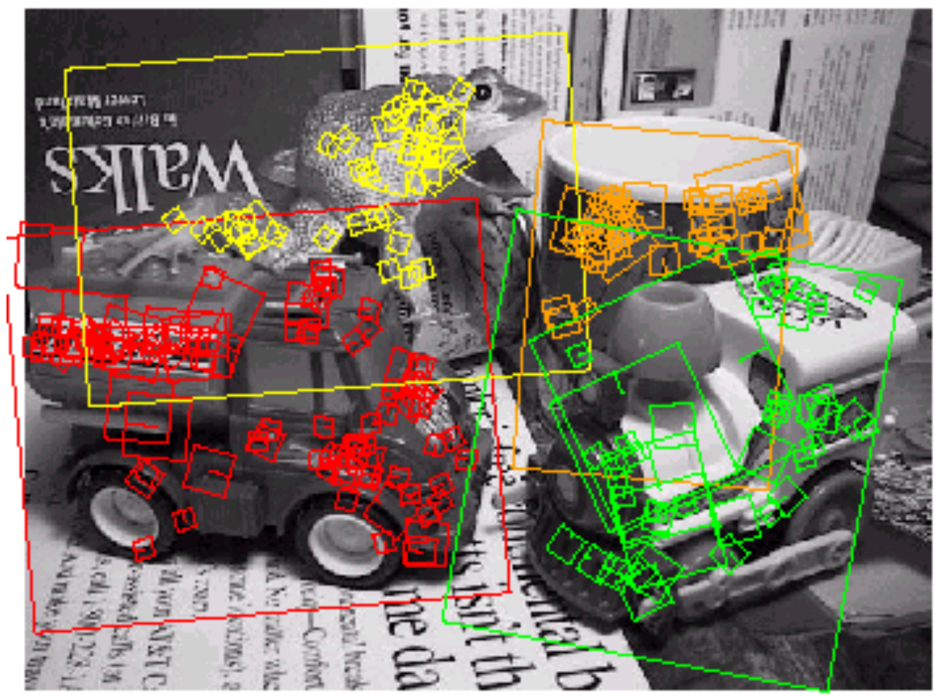
$$\mathbf{x} = [\mathbf{A}^{\mathrm{T}}\mathbf{A}]^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Finding the objects (in detail)

Source: D. Hoeim

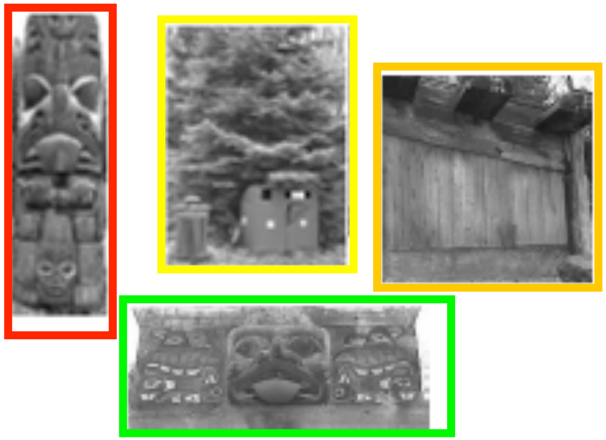
1. Match interest points from input image to database image
2. Get location/scale/orientation using Hough voting
   - In training, each point has known position/scale/orientation wrt whole object
   - Matched points vote for the position, scale, and orientation of the entire object
   - Bins for x, y, scale, orientation
     - Wide bins (0.25 object length in position, 2x scale, 30 degrees orientation)
     - Vote for two closest bin centers in each direction (16 votes total)
3. Geometric verification
   - For each bin with at least 3 keypoints
   - Iterate between least squares fit and checking for inliers and outliers
4. Report object if > T inliers (T is typically 3, can be computed to match some probabilistic threshold)

Examples of recognized objects

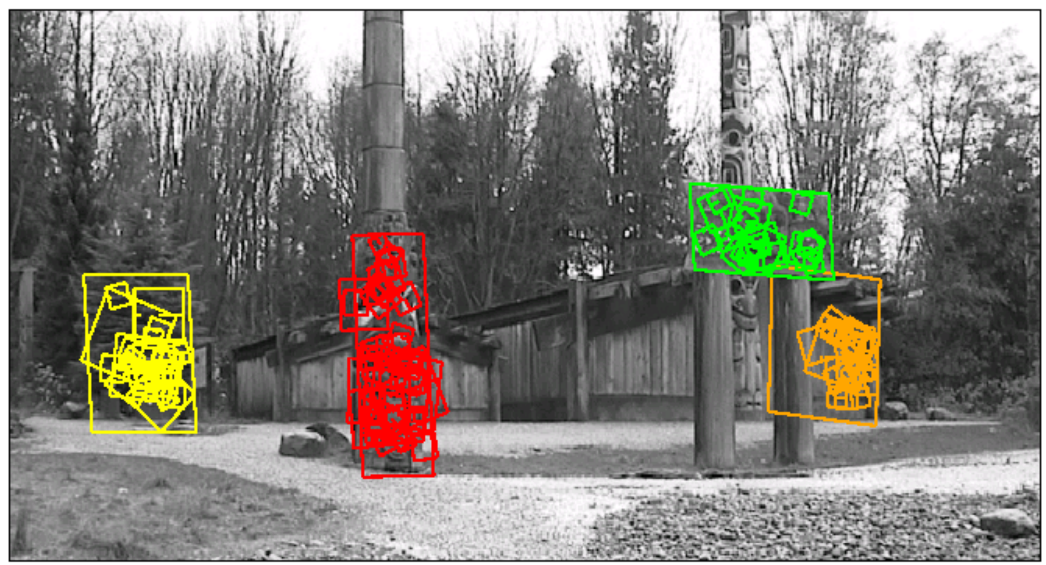


Slide credit: David Lowe

Location Recognition



Training



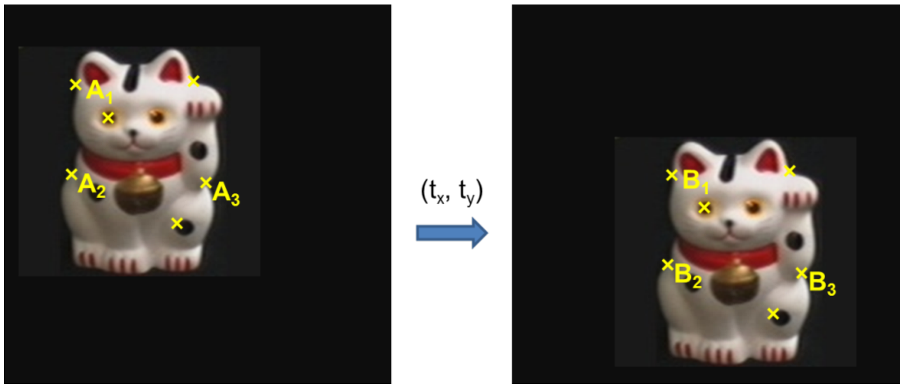
[Lowe04]

Slide credit: David Lowe

Key concepts

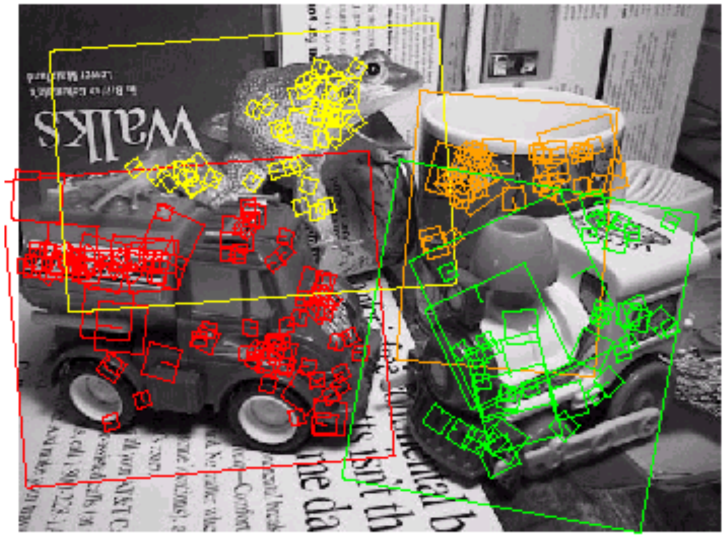
Alignment as robust fitting

- Affine transformations
- Homographies
- Descriptor-based feature matching
- RANSAC



Object instance recognition

- Find keypoints, compute descriptors
- Match descriptors
- Vote for / fit affine parameters
- Return object if # inliers > T



Adapted from D. Hoeim