Computer Vision & Image Processing
CSE 473 / 573

Instructor - Kevin R. Keane, PhD

TAs - Radhakrishna Dasari, Yuhao Du, Niyazi Sorkunlu

Lecture 26

October 30, 2017

Image Alignment

Image Alignment

Readings for today:

- Forsyth and Ponce chapter 12

Image alignment

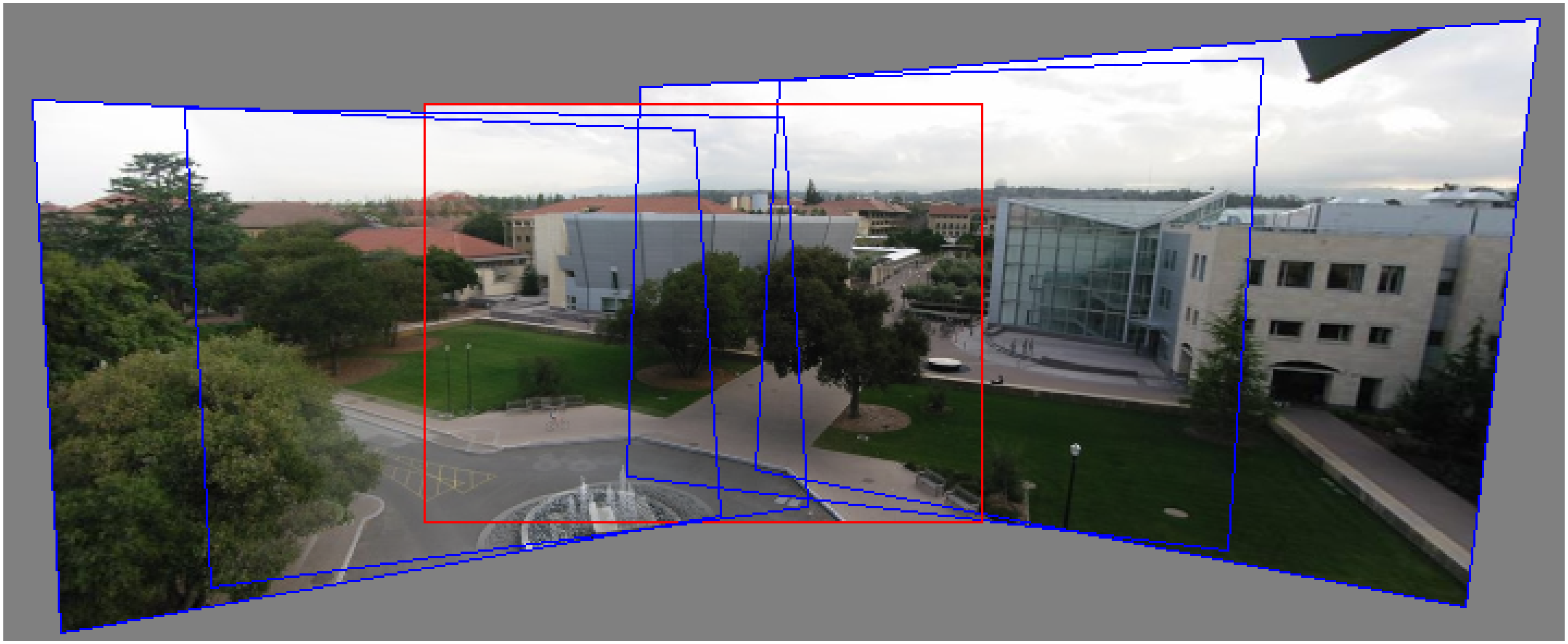
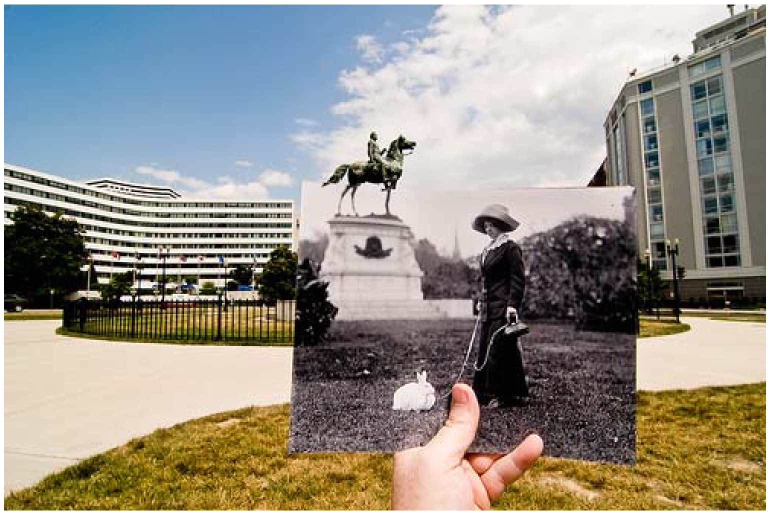
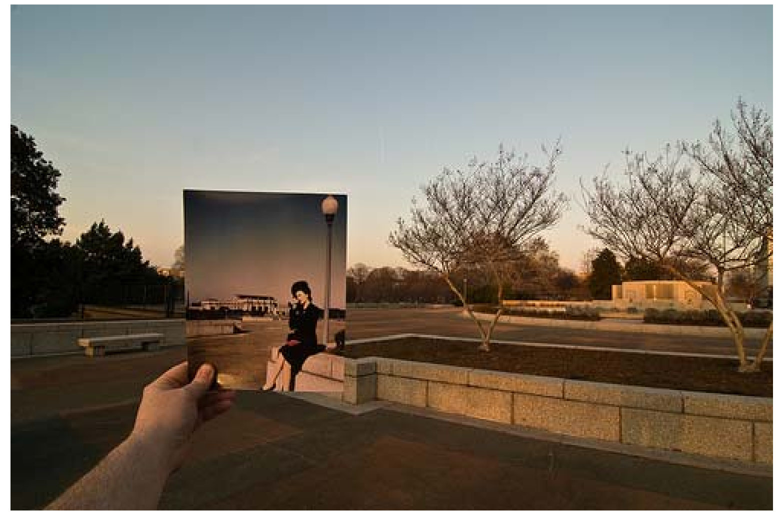


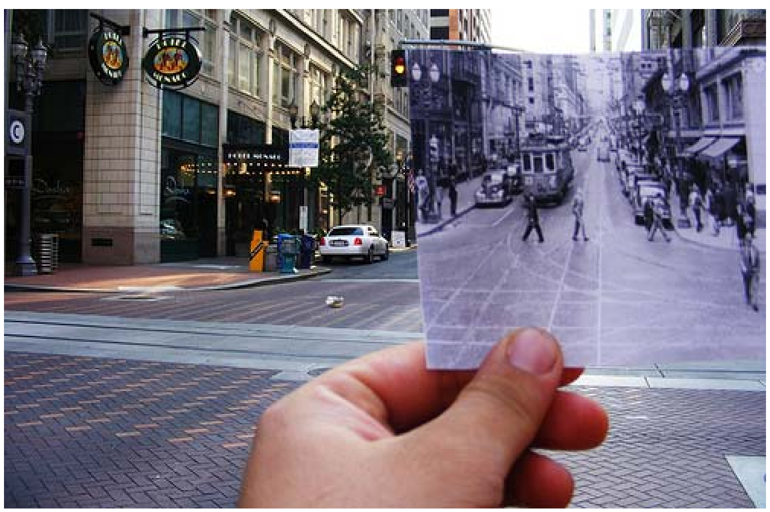
Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fall/

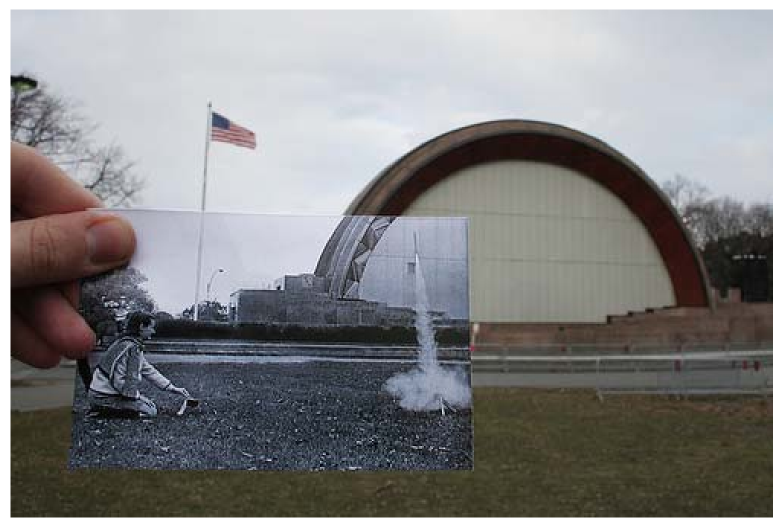
Alignment applications

- [A look into the past]









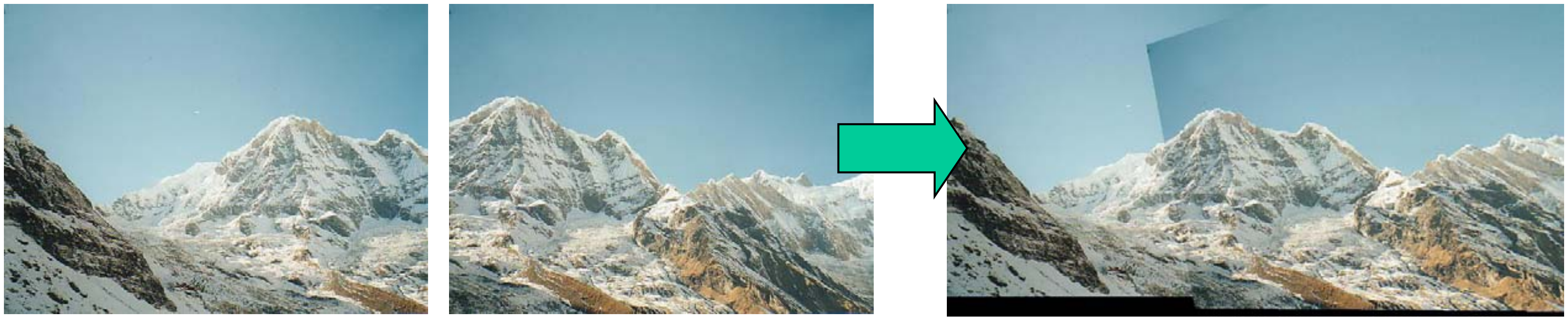
Alignment applications

- A look into the past

Alignment applications

- [Streetside images]()

Alignment applications



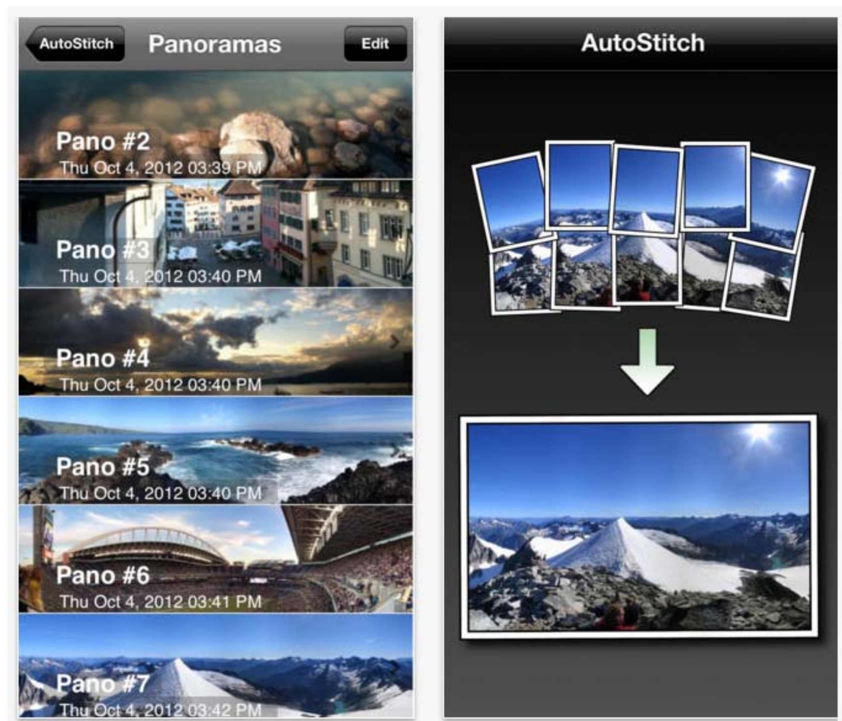
Panorama stitching

AutoStitch Panorama

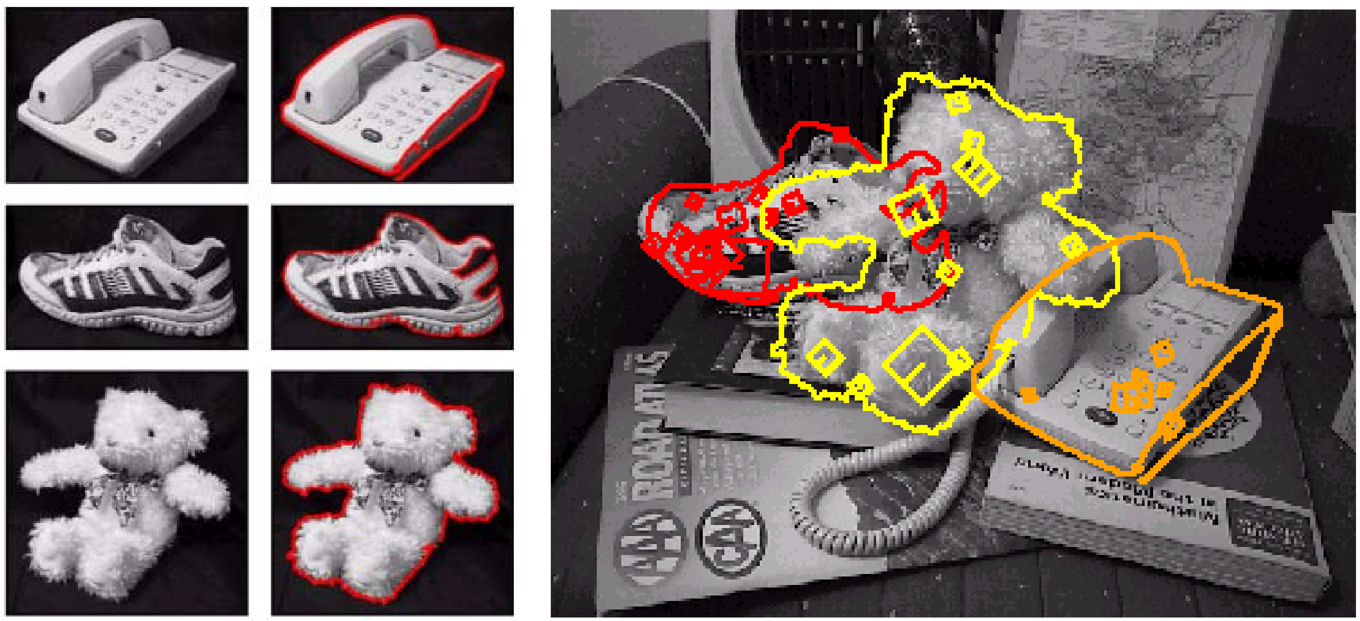
By Cloudburst Research Inc.

Open iTunes to buy and download apps.

View In iTunes

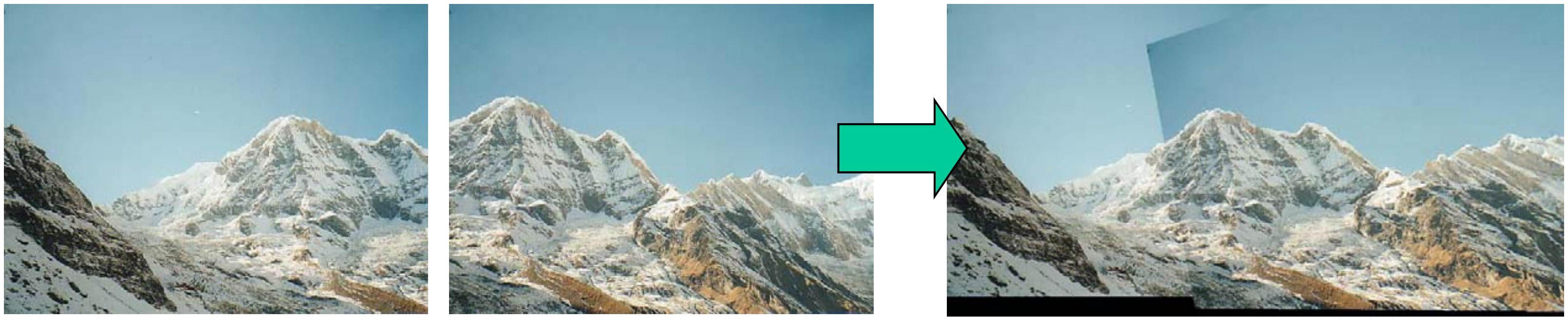


Alignment applications



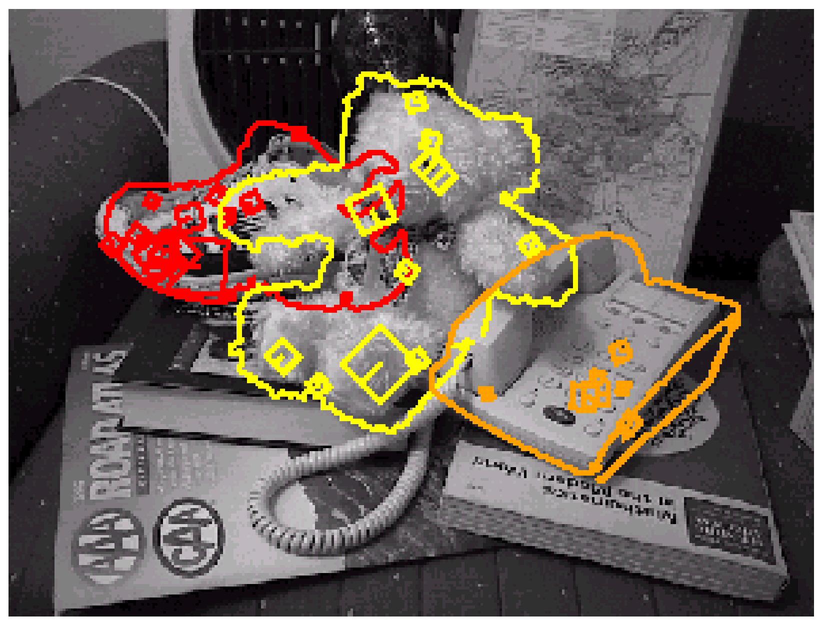
Recognition of object instances

Alignment challenges



Small degree of overlap

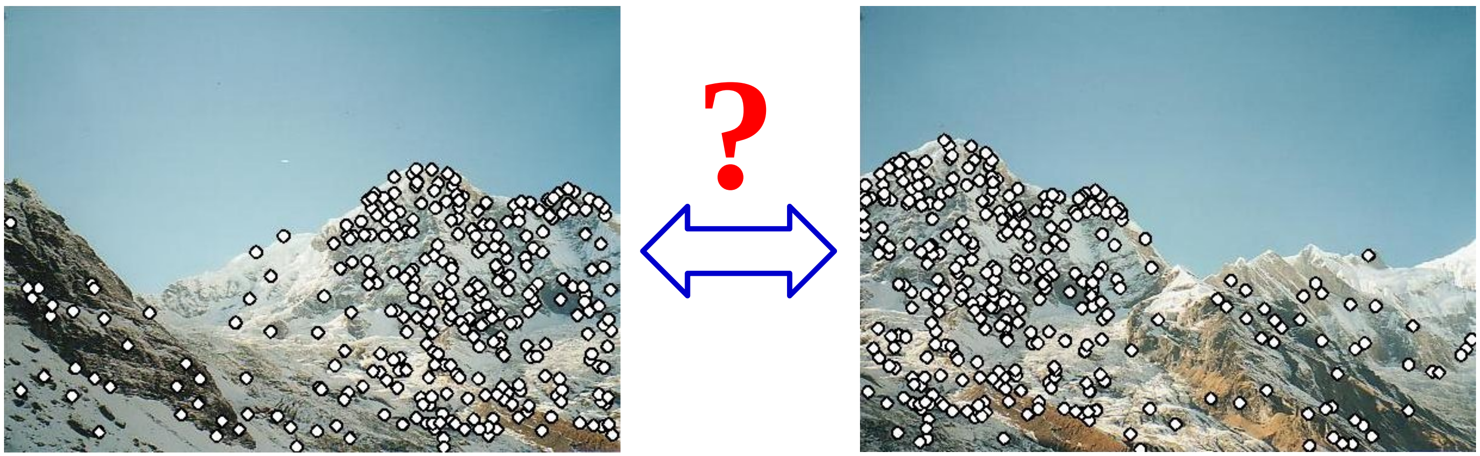
Intensity changes



Occlusion,
clutter

Feature-based alignment

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

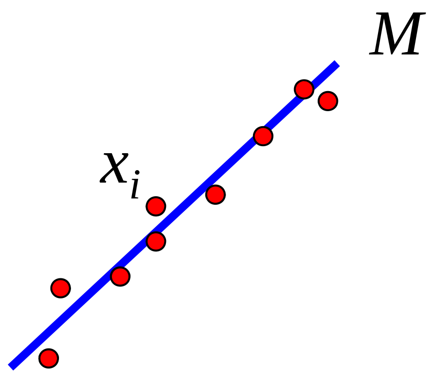


Feature-based alignment: Overview

- Alignment as fitting
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC
- Large-scale alignment
  - Inverted indexing
  - Vocabulary trees
- Application: searching the night sky

Alignment as fitting

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

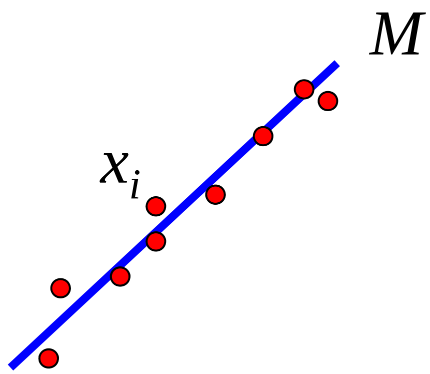


Find model M that minimizes

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

Alignment as fitting

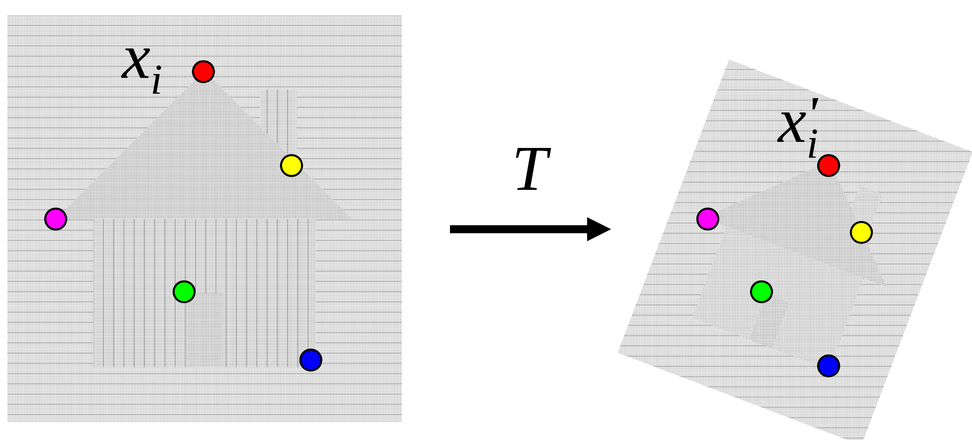
- Previous lectures: fitting a model to features in one image



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

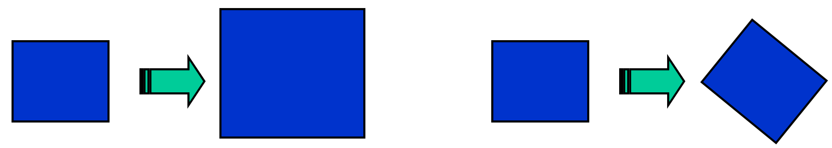


Find transformation T that minimizes

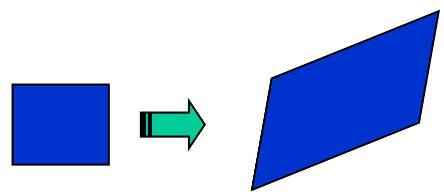
$$\sum_i \text{residual}(T(x_i), x_i')$$

2D transformation models

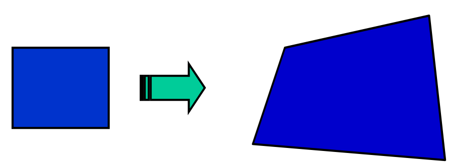
- Similarity (translation, scale, rotation)



- Affine

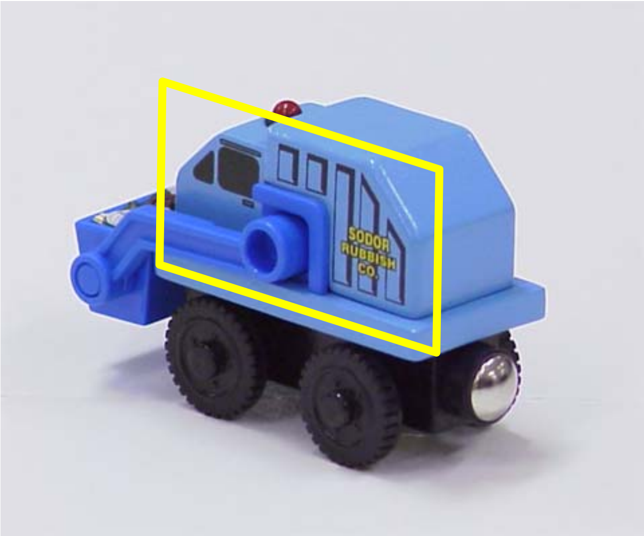


- Projective (homography)



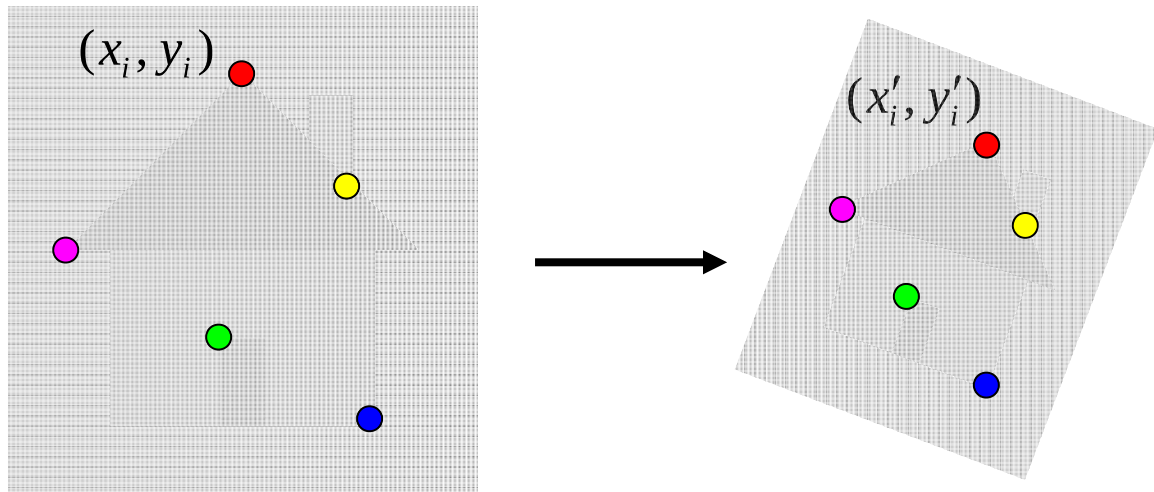
Let’s start with affine transformations

- 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?



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

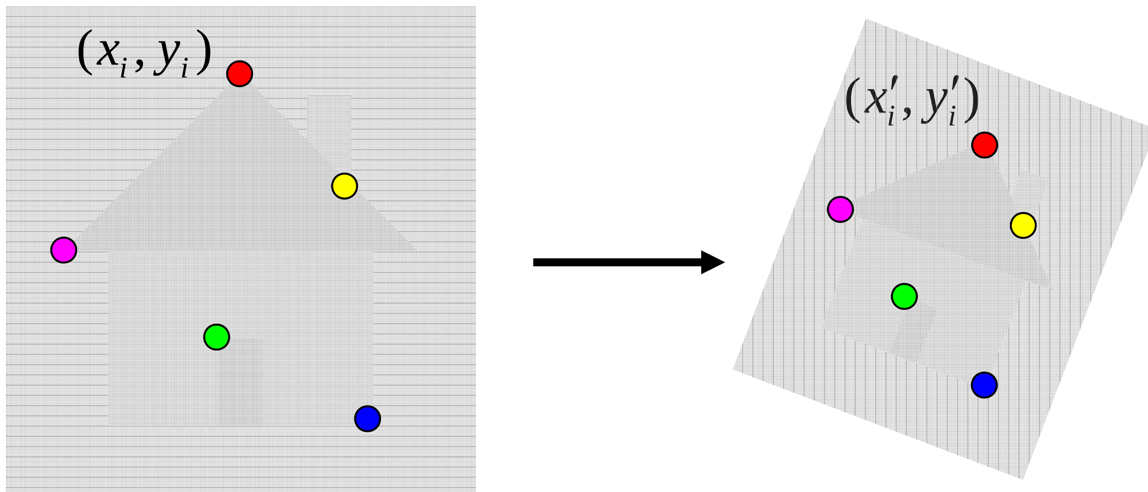
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

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

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

Fitting an affine transformation

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



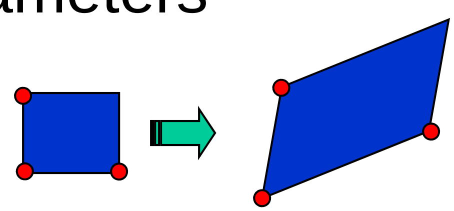
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

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

Fitting an affine transformation

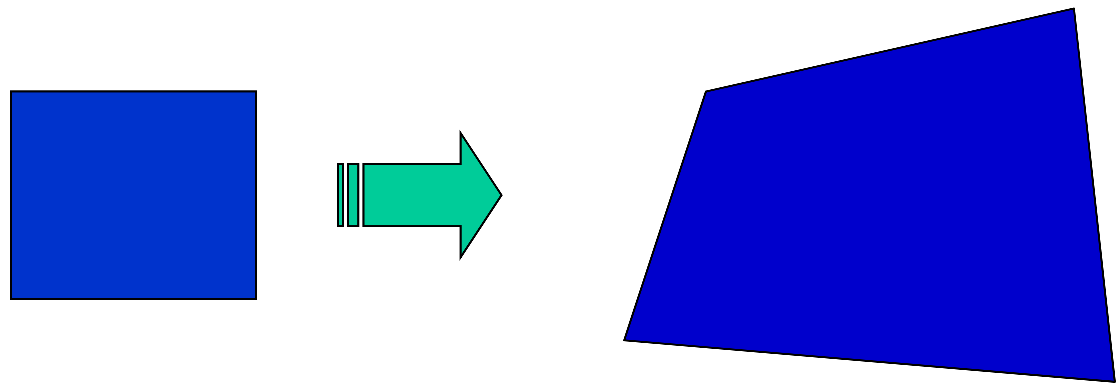
$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

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



Fitting a plane projective transformation

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



Homography

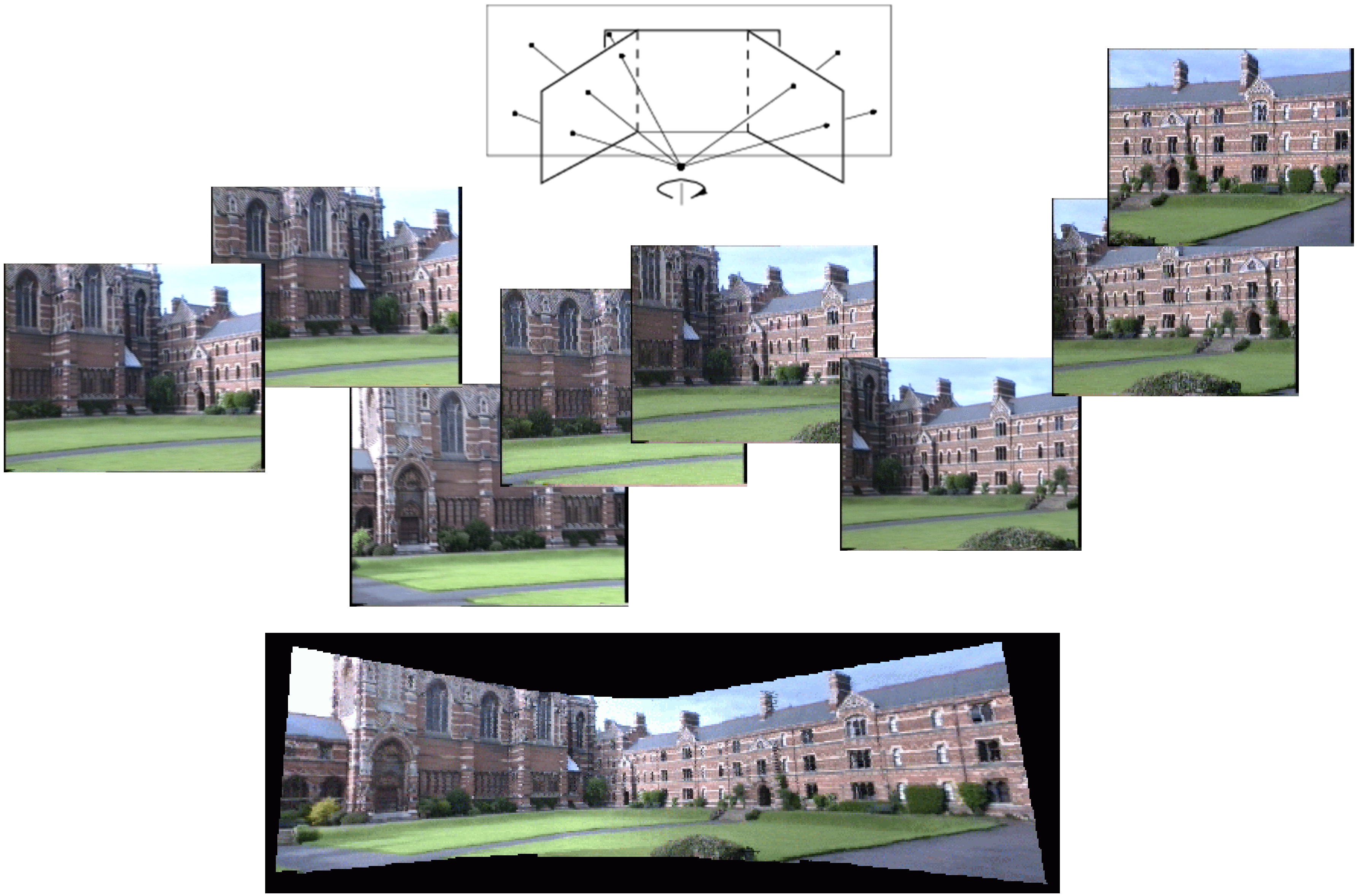
- The transformation between two views of a planar surface

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

Application: Panorama stitching



Source: Hartley & Zisserman

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

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:

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

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

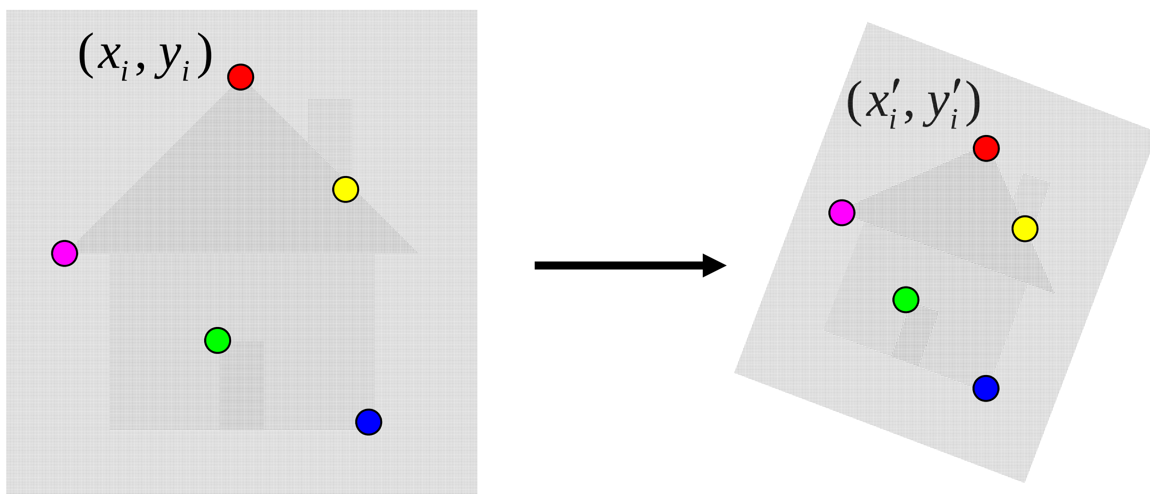
Fitting a homography

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

- 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 $\|\mathbf{Ah}\|^2$
  - Eigenvector of $\mathbf{A}^\mathbf{T}\mathbf{A}$ corresponding to smallest eigenvalue
  - Four matches needed for a minimal solution

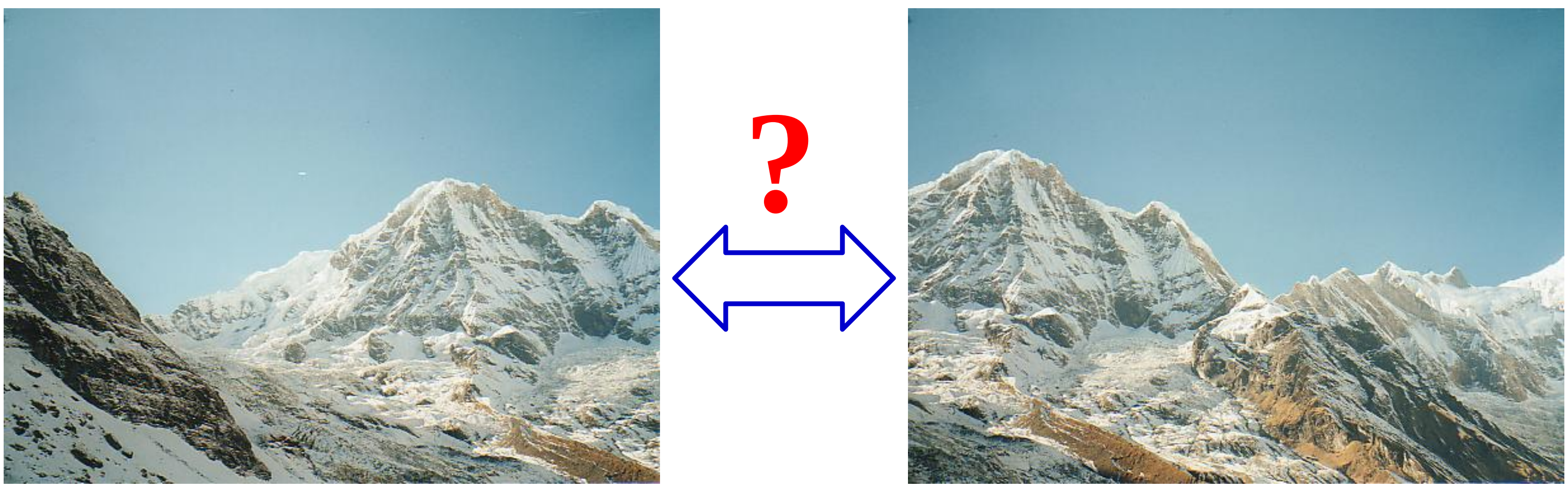
Robust feature-based alignment

- 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?



Robust feature-based alignment

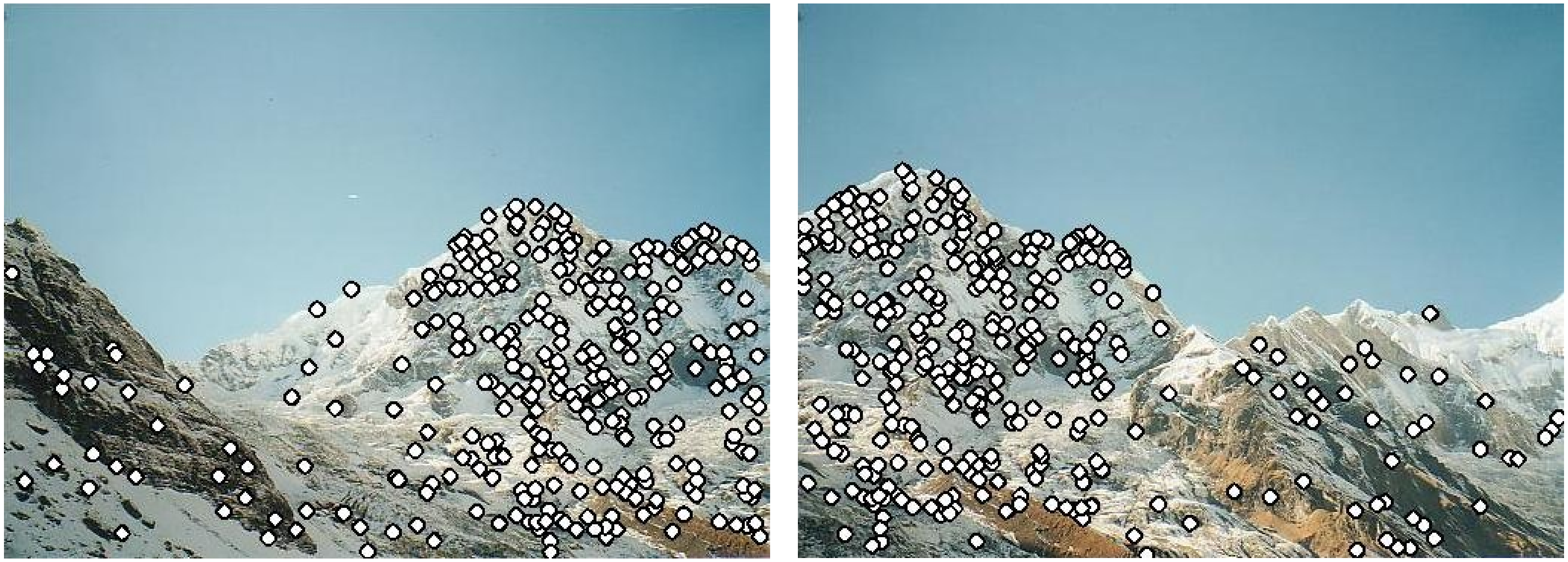
- So far, we've assumed that we are given a set of "ground-truth" correspondences between the two images we want to align
- What if we don't know the correspondences?



Robust feature-based alignment

Robust feature-based alignment

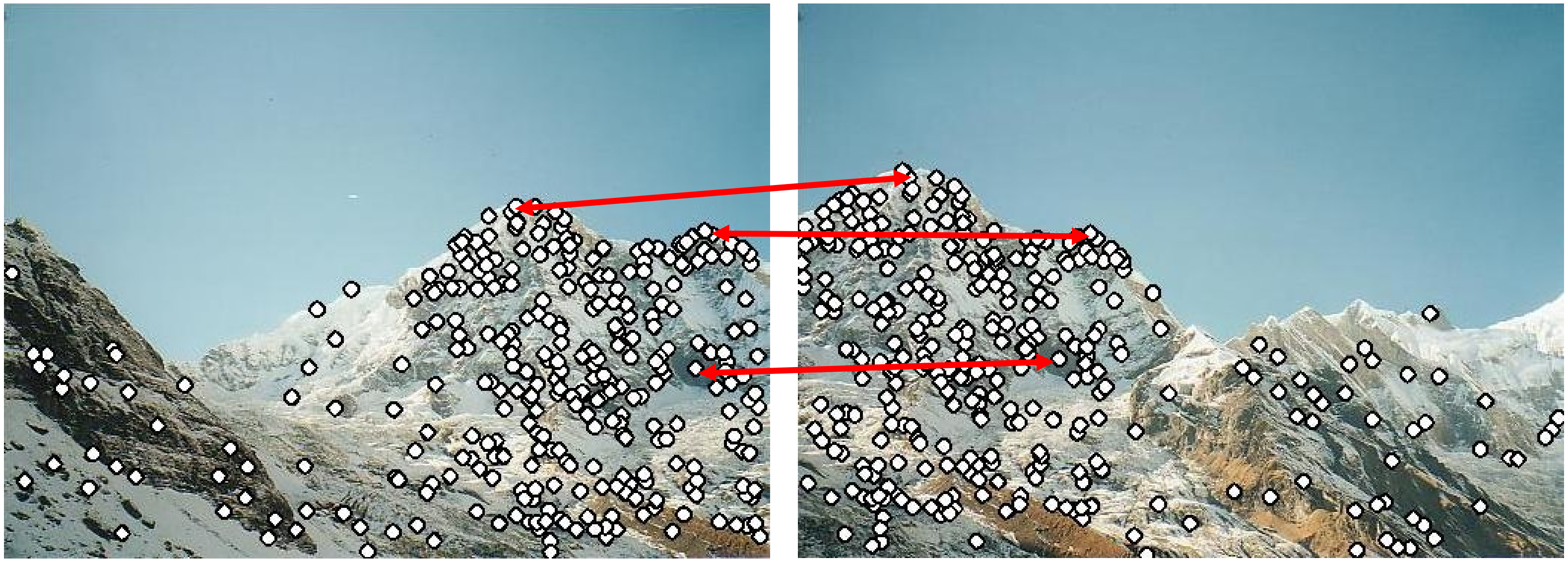


- Extract features

Robust feature-based alignment

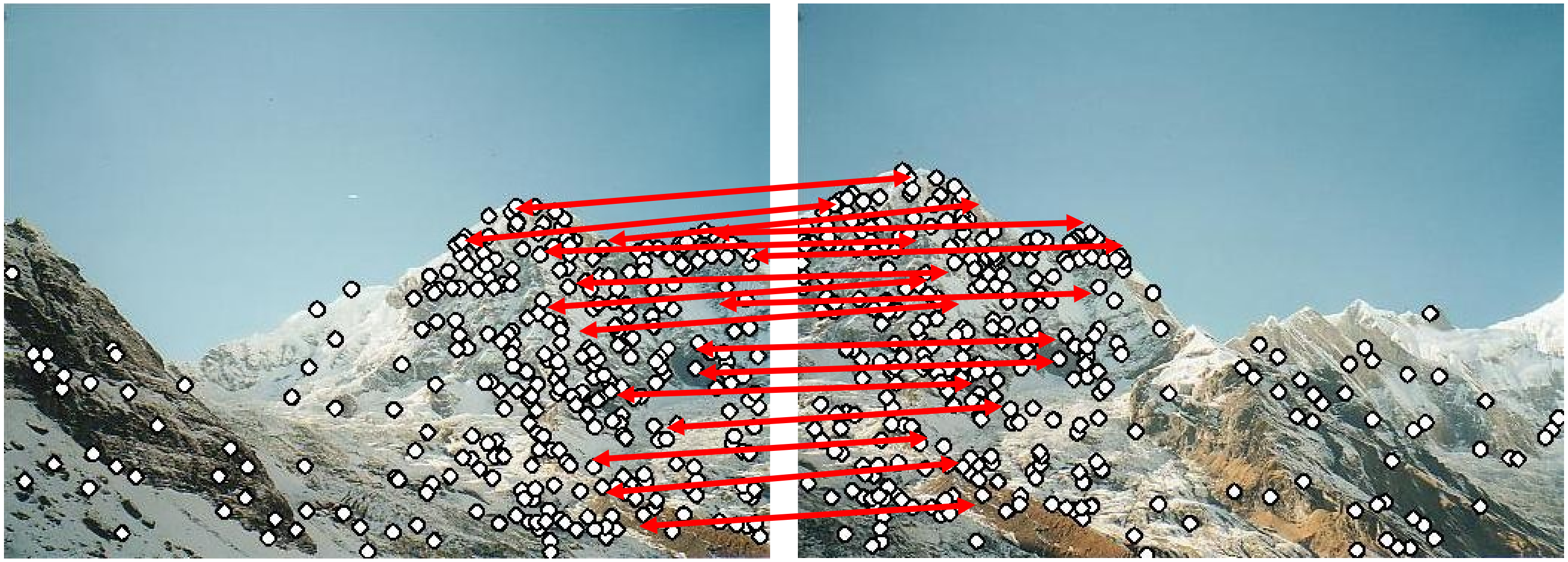
- Extract features
- Compute *putative matches*

Robust feature-based alignment



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

Robust feature-based alignment

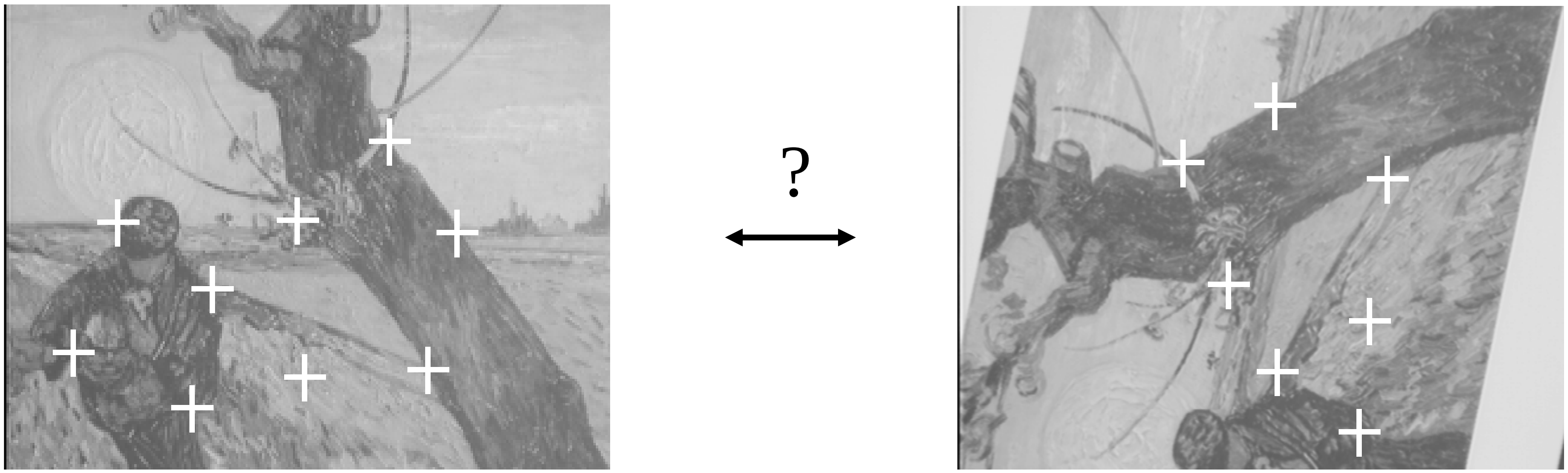


- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T*
  - *Verify* transformation (search for other matches consistent with *T*)

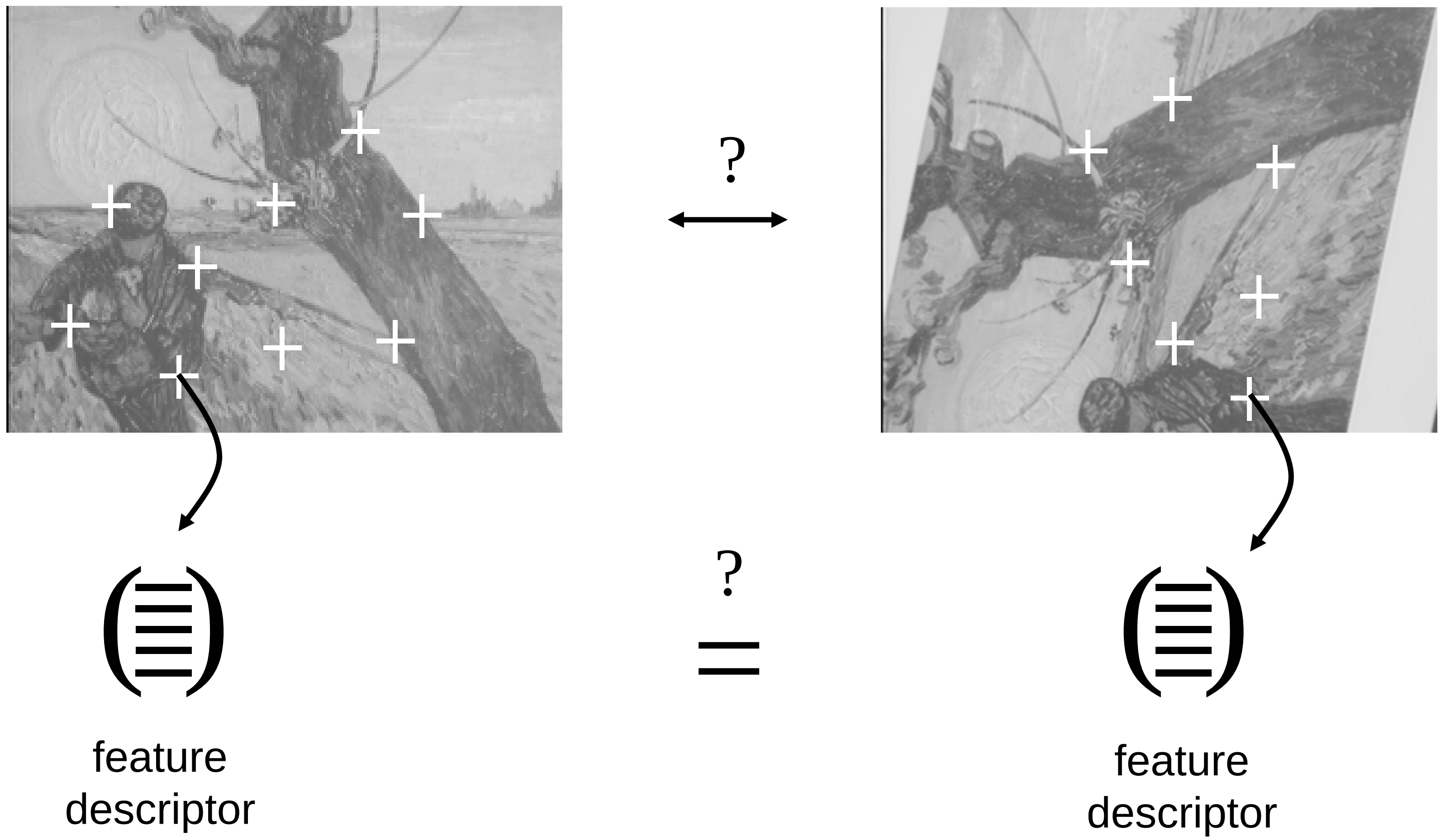
Robust feature-based alignment

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

Generating putative correspondences



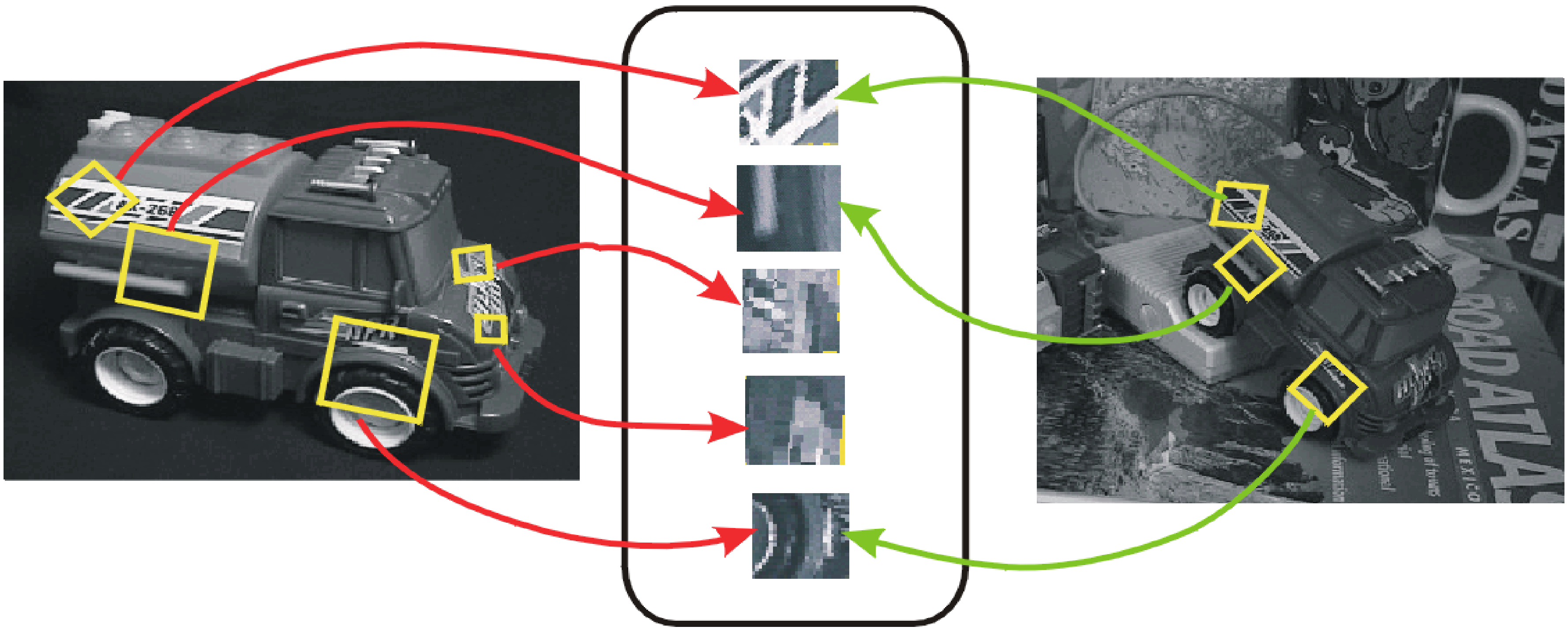
Generating putative correspondences



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

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

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

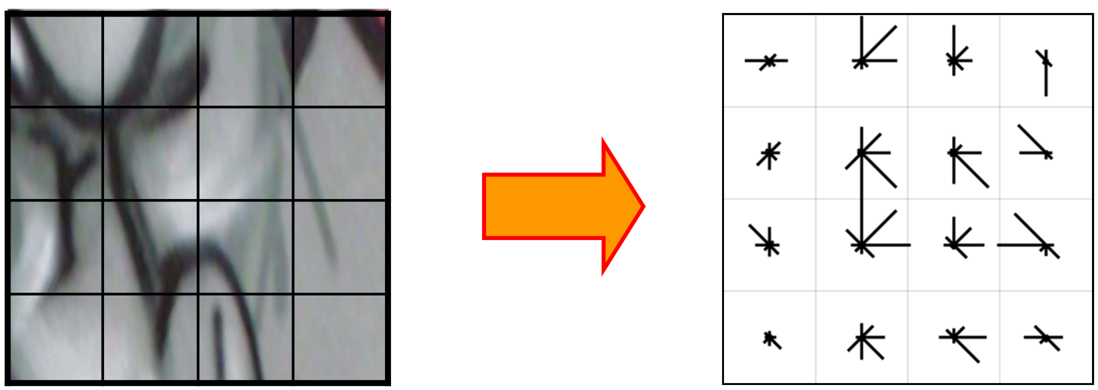
    – Invariant to affine intensity change

Disadvantage of intensity vectors as descriptors

- Small deformations can affect the matching score a lot

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.

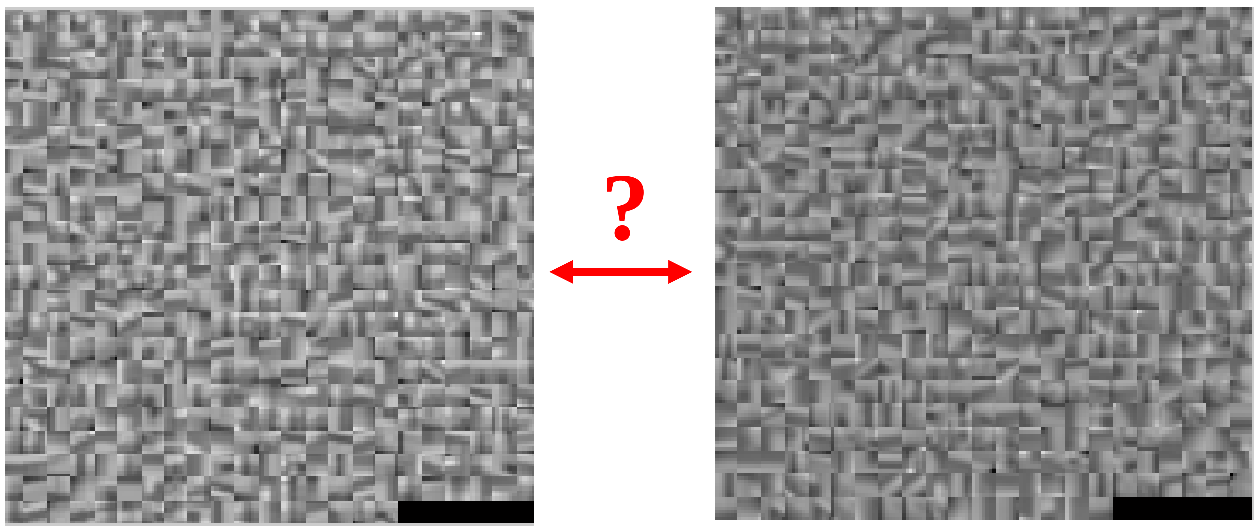
Feature descriptors: SIFT

- 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

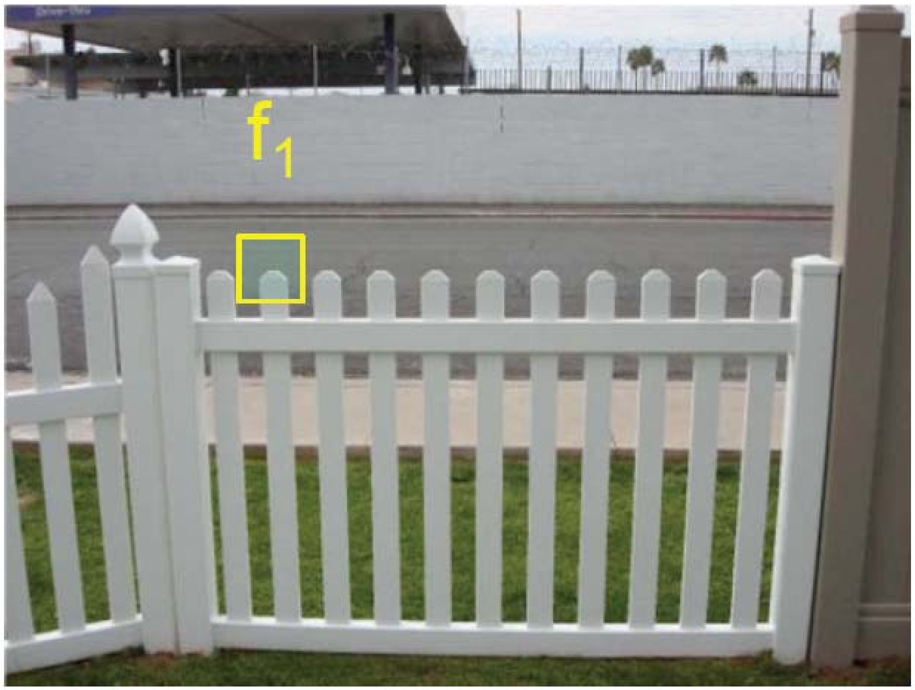
David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

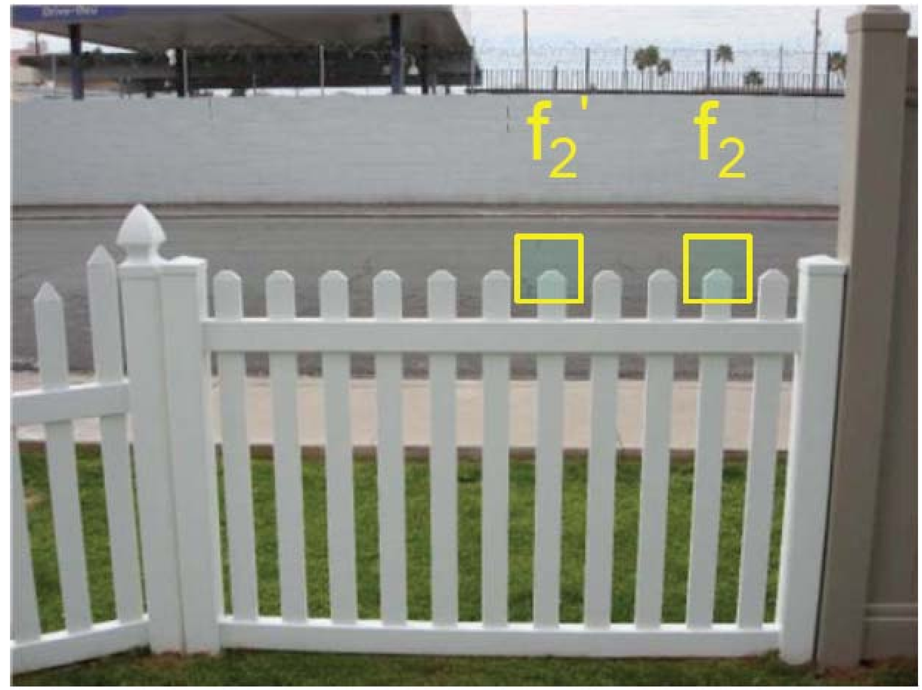
Feature matching

- 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



Problem: Ambiguous putative matches

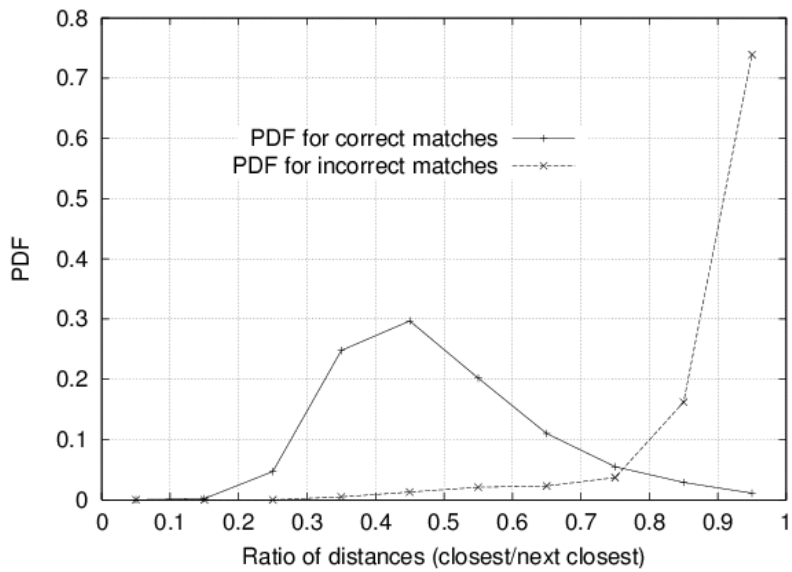




Source: Y. Furukawa

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.

RANSAC

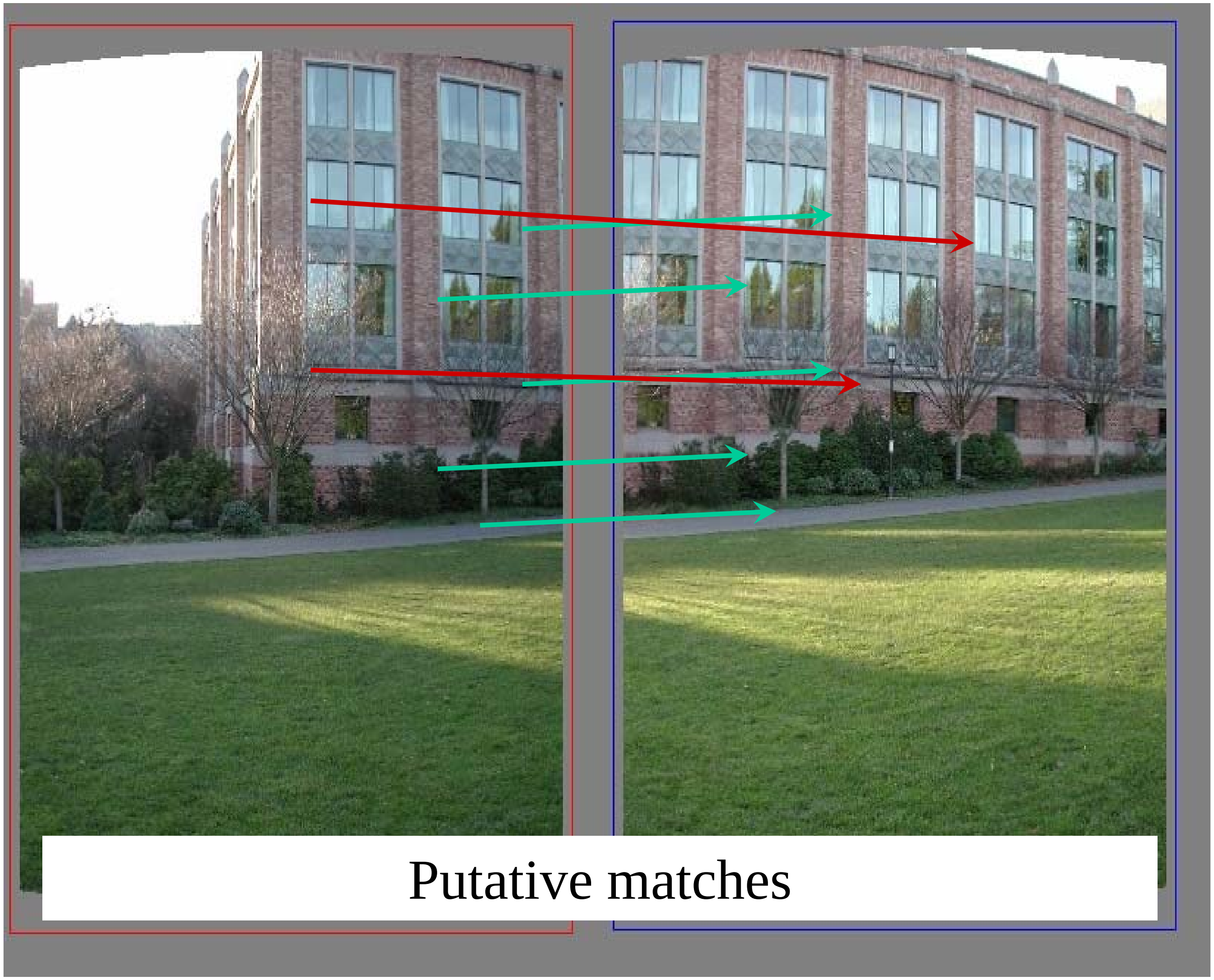
- The set of putative matches contains a very high percentage of outliers

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

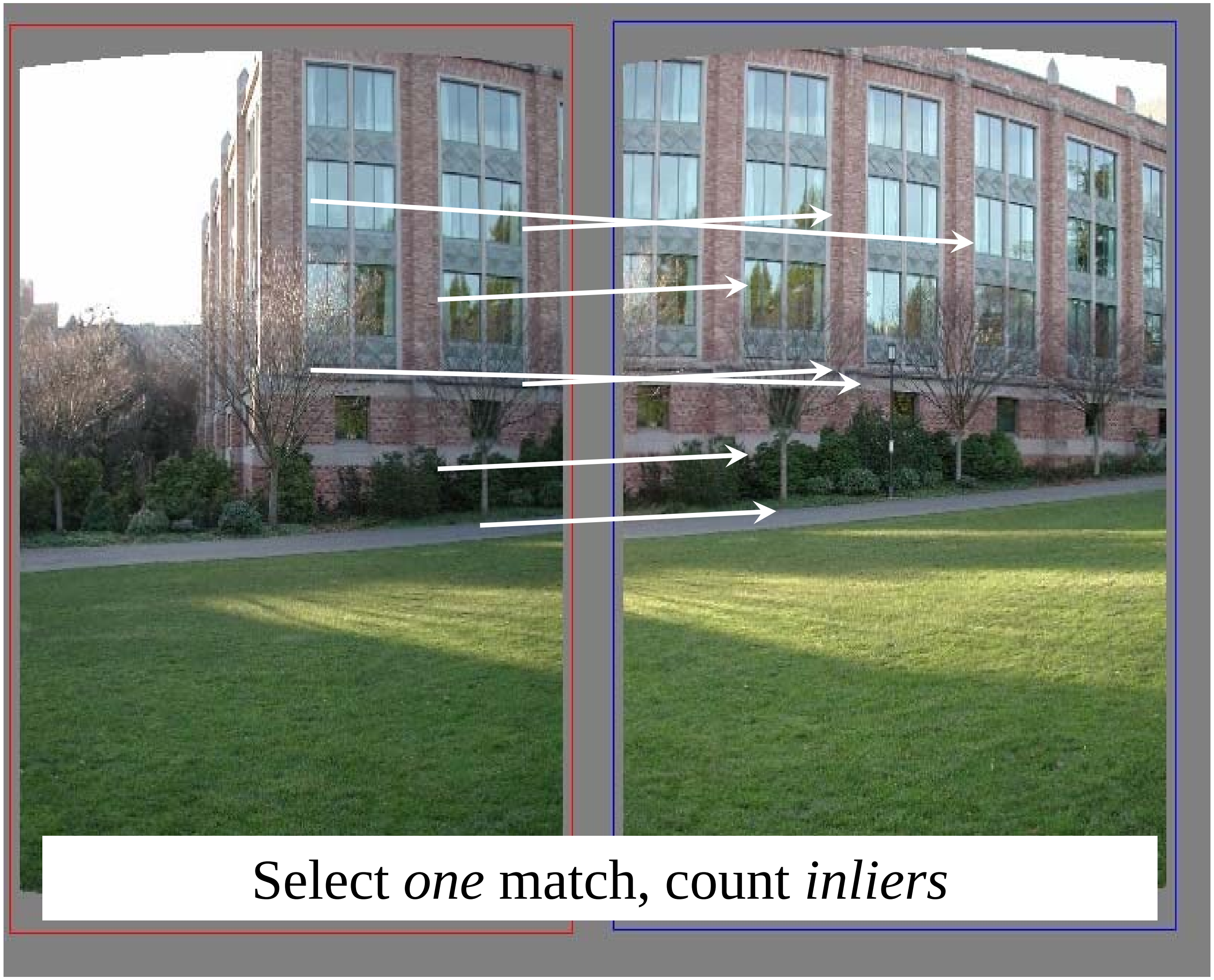
Keep the transformation with the largest number of inliers

RANSAC example: Translation



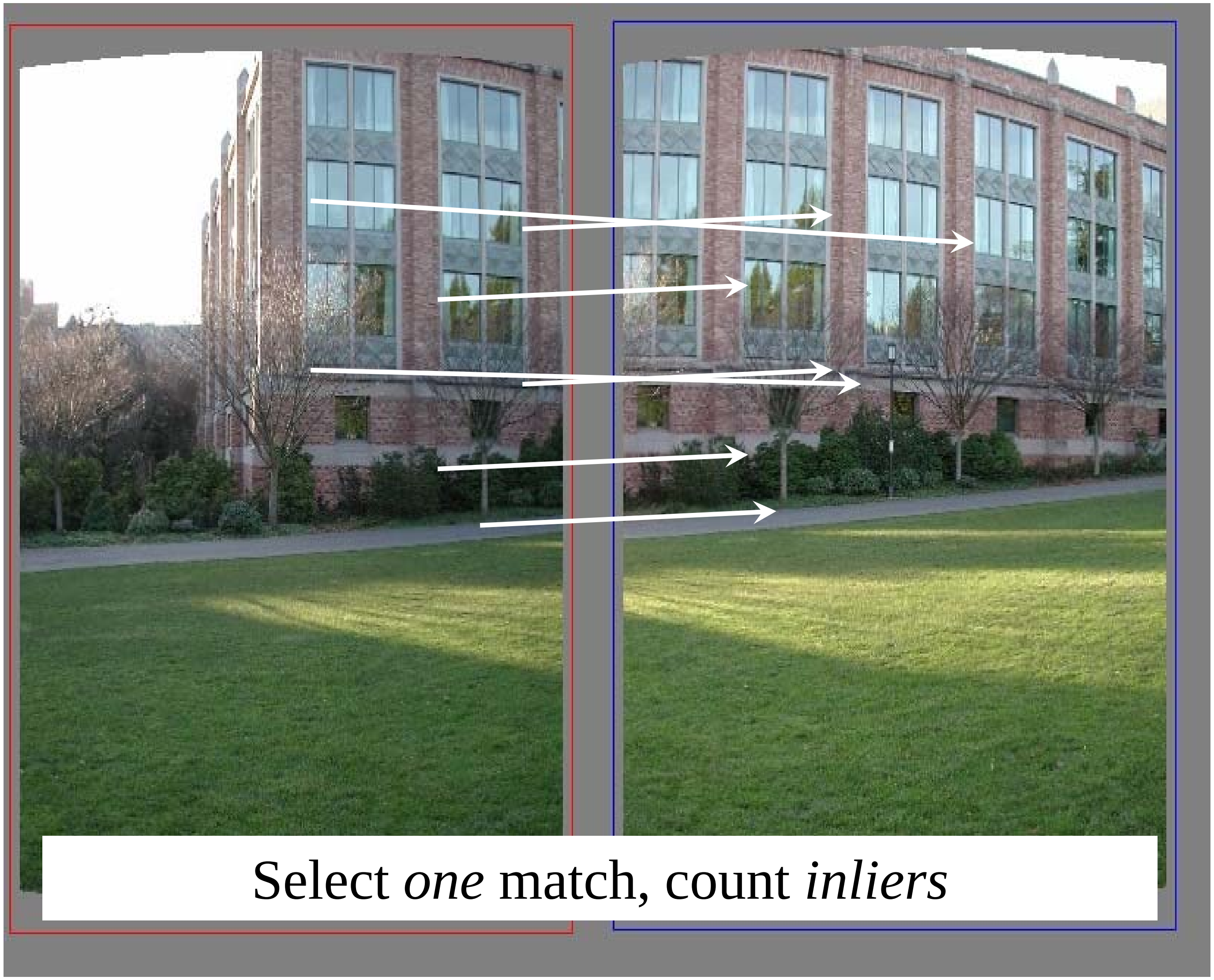
Putative matches

RANSAC example: Translation

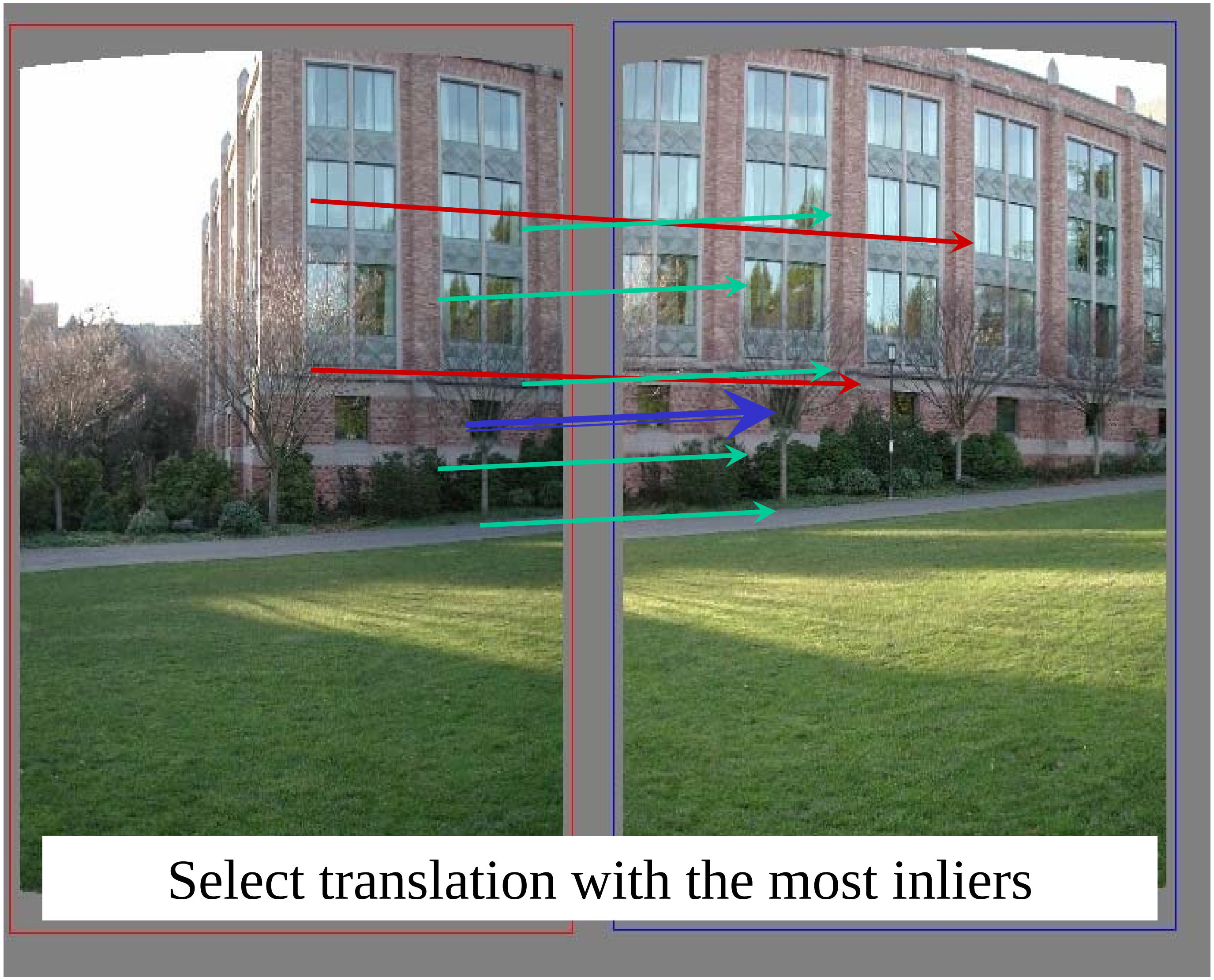


Select *one* match, count *inliers*

RANSAC example: Translation



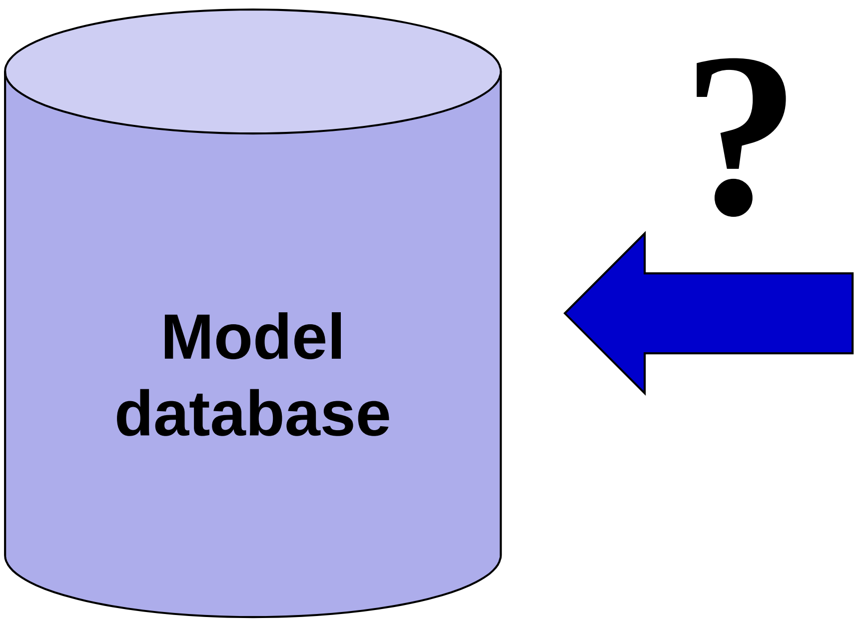
RANSAC example: Translation



Select translation with the most inliers

Scalability: Alignment to large databases

- What if we need to align a test image with thousands or millions of images in a model database?
  - Efficient putative match generation
    - Approximate descriptor similarity search, inverted indices



Test image

Large-scale visual search

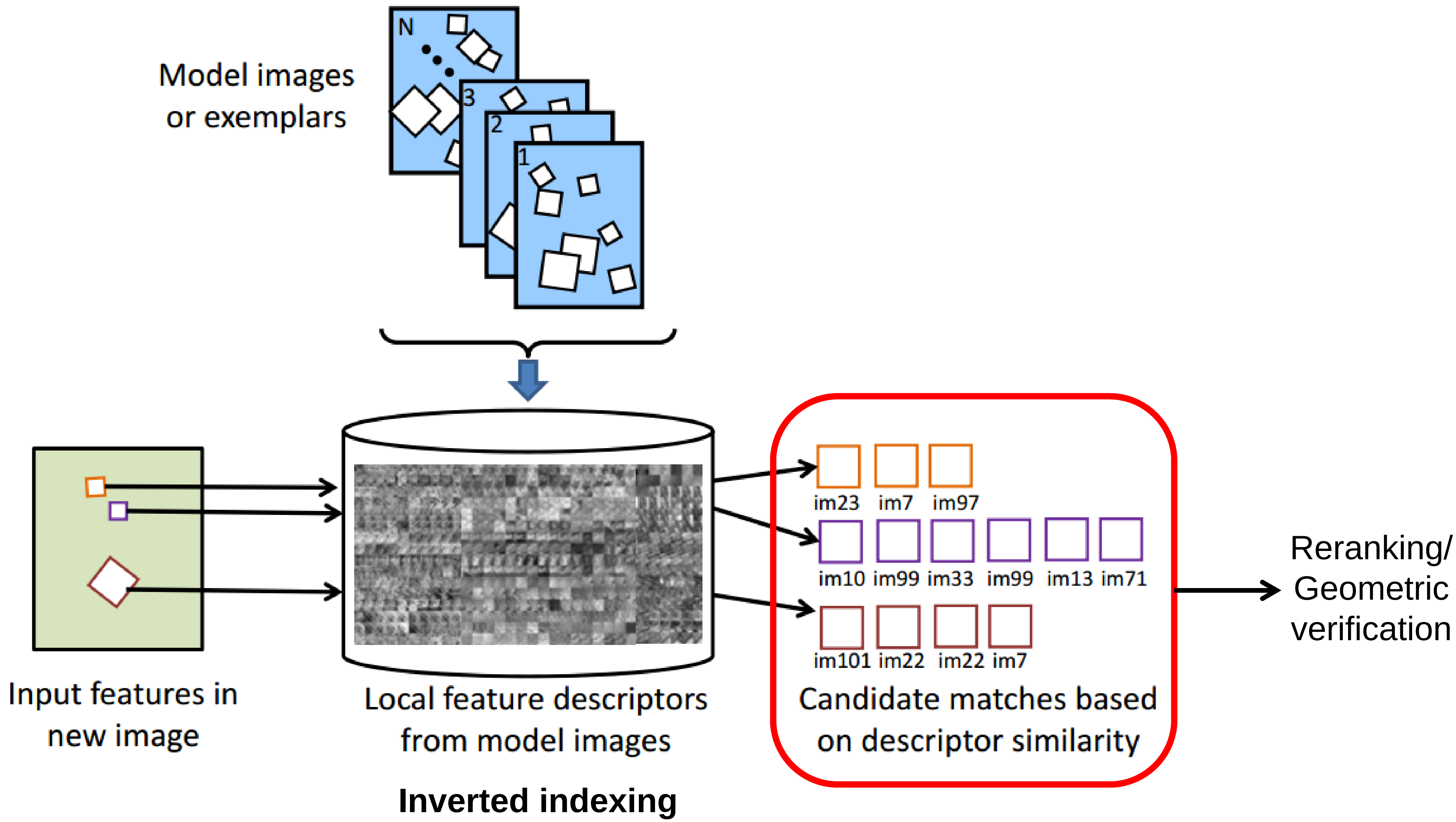
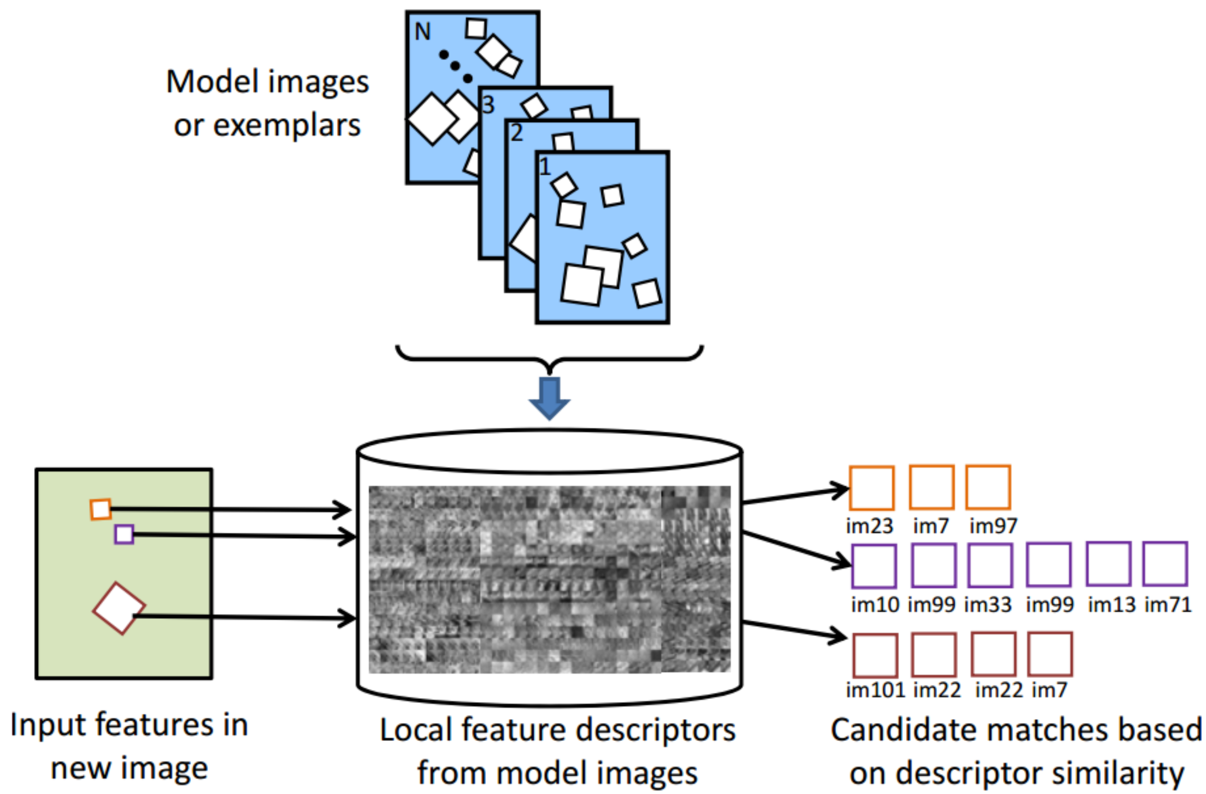


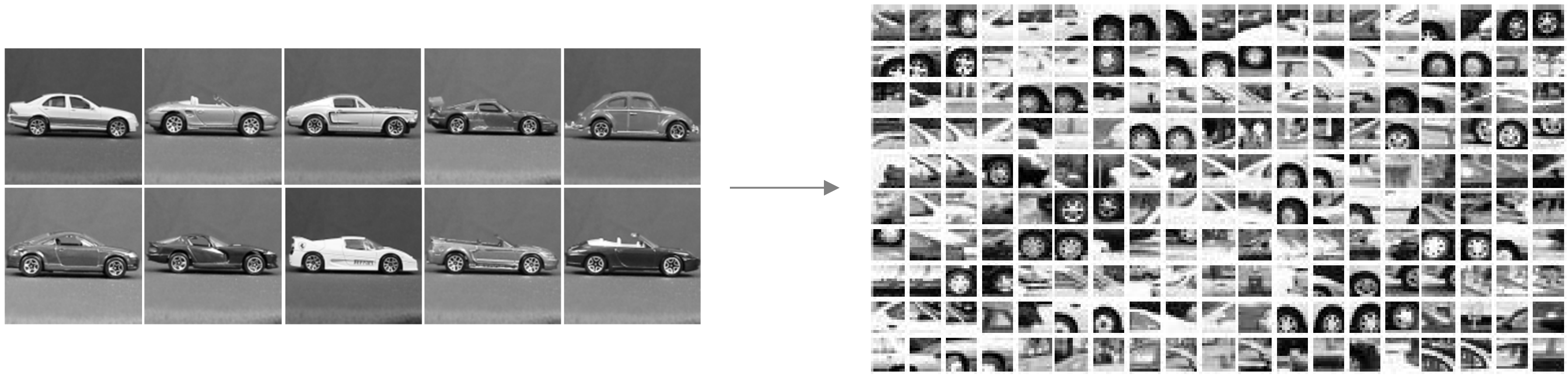
Figure from: Kristen Grauman and Bastian Leibe, Visual Object Recognition, Synthesis Lectures on Artificial Intelligence and Machine Learning, April 2011, Vol. 5, No. 2, Pages 1-181

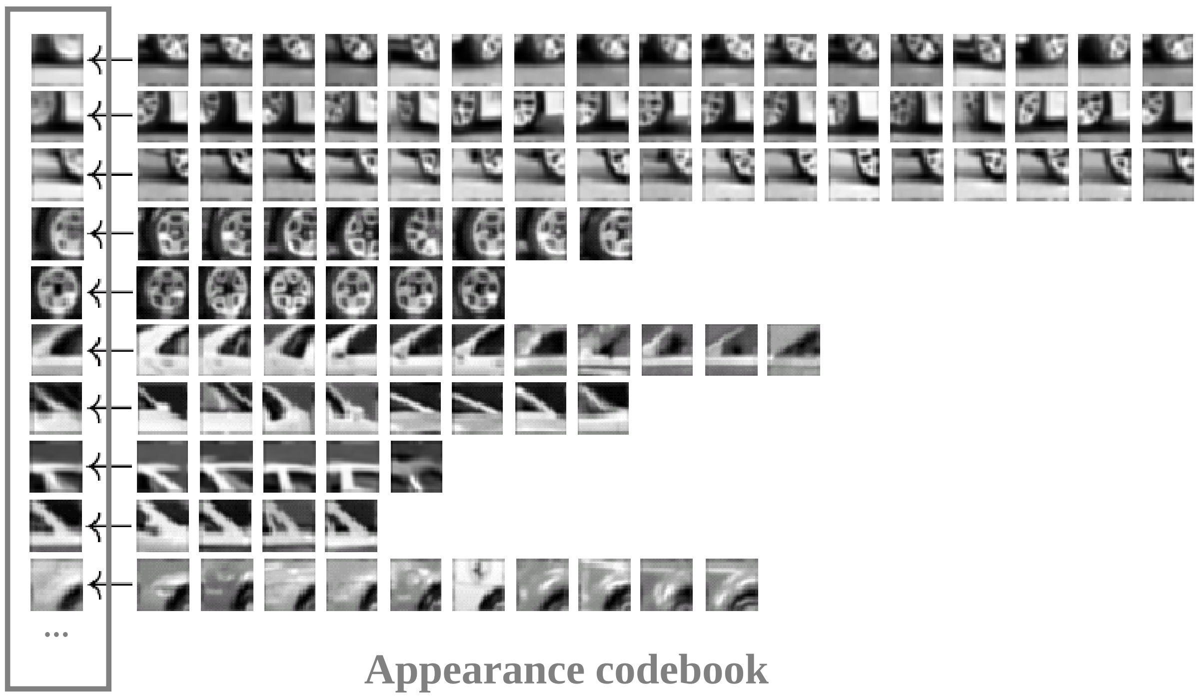
How to do the indexing?



- Idea: find a set of *visual codewords* to which descriptors can be *quantized*

Recall: Visual codebook for generalized Hough transform





Appearance codebook

Source: B. Leibe

K-means clustering

- Want to minimize sum of squared Euclidean distances between points $\mathbf{x}_i$ and their nearest cluster centers $\mathbf{m}_k$

$$D(X, M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (\mathbf{x}_i - \mathbf{m}_k)^2$$

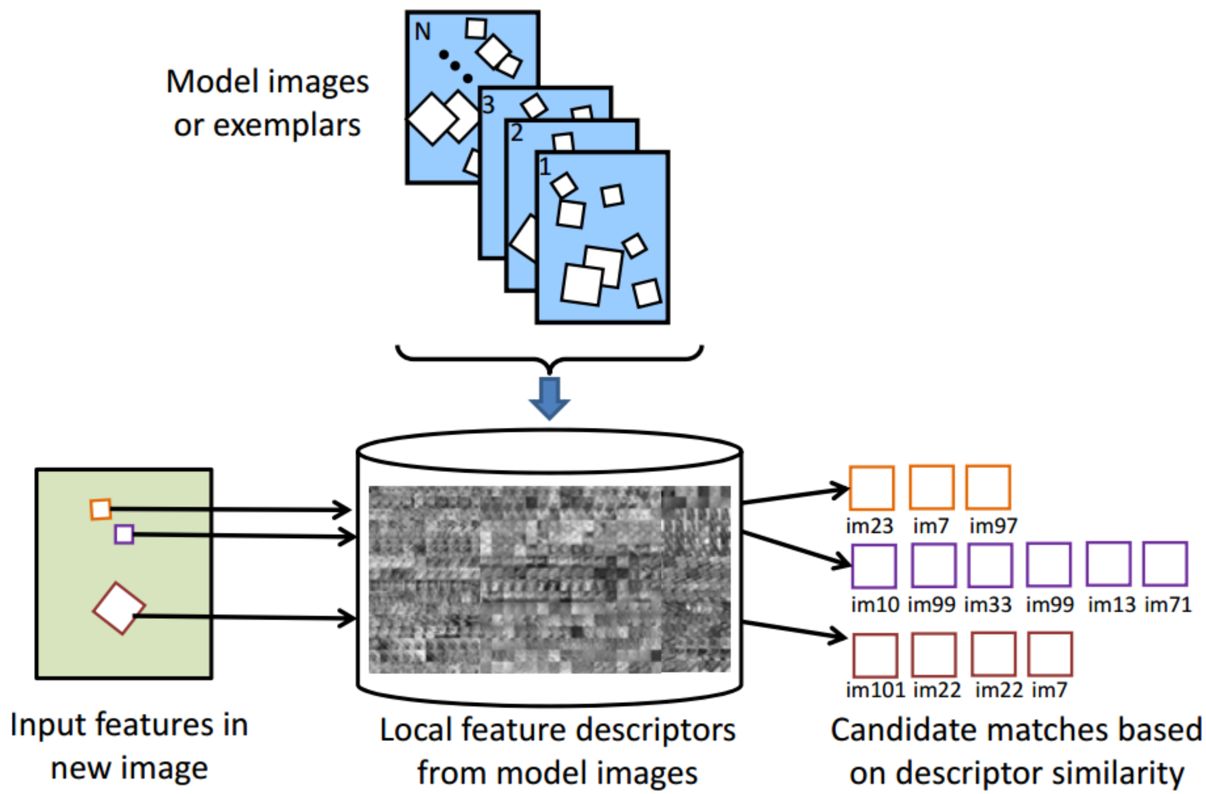
Algorithm:

- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

K-means demo

Source: http://shabal.in/visuals/kmeans/1.html
Another demo: http://www.kovan.ceng.metu.edu.tr/~maya/kmeans/

How to do the indexing?



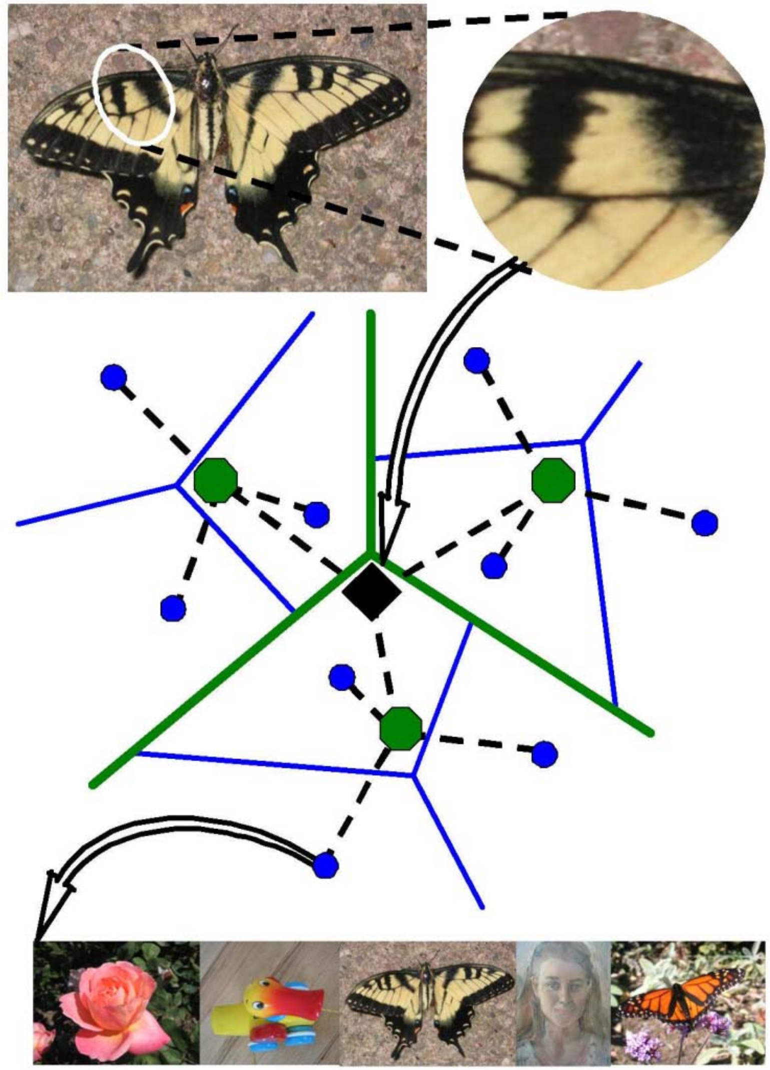
- Cluster descriptors in the database to form codebook
- At query time, quantize descriptors in query image to nearest codevectors
- Problem solved?

Efficient indexing technique: Vocabulary trees

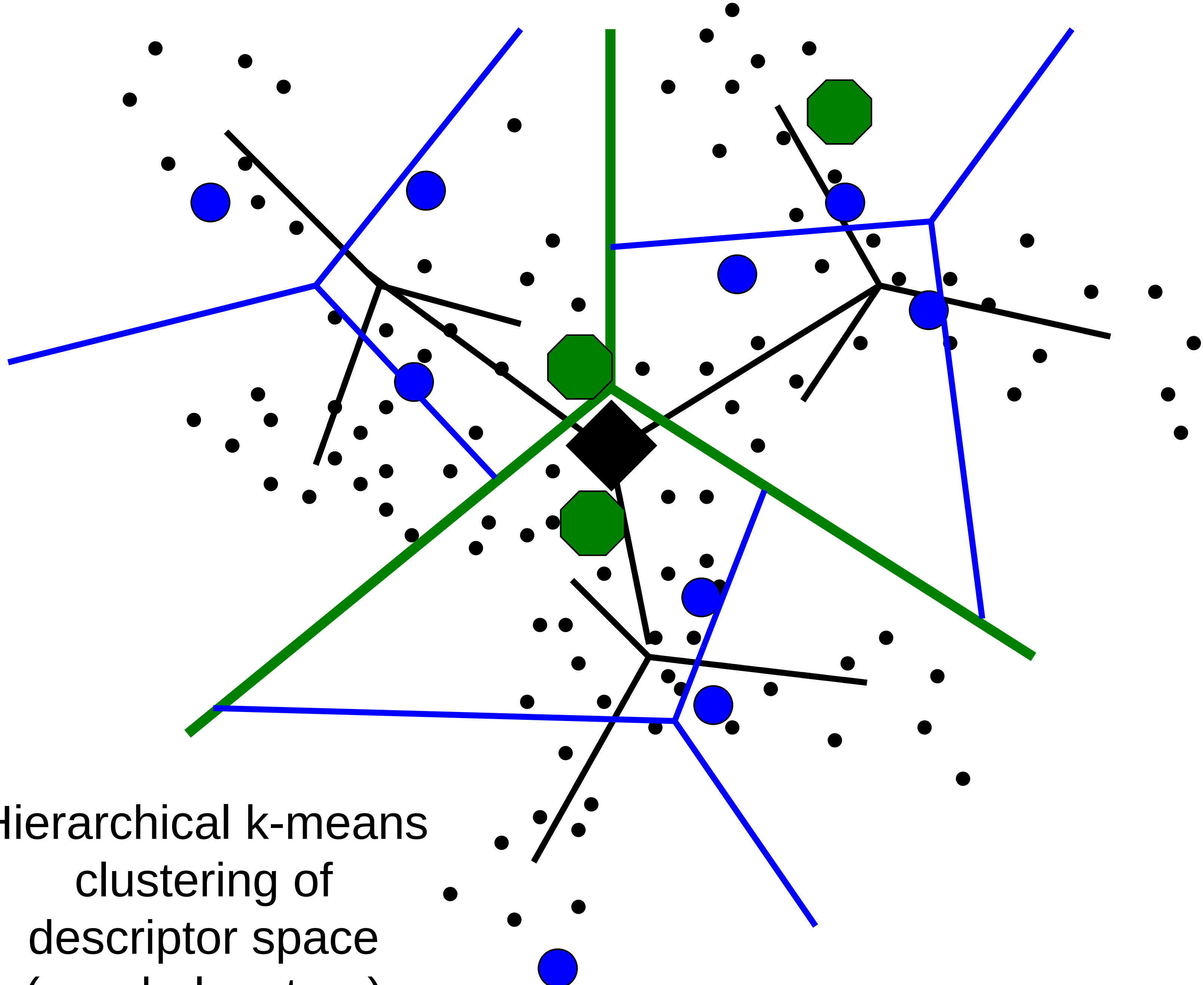
Test image

Vocabulary tree
with inverted
index

Database

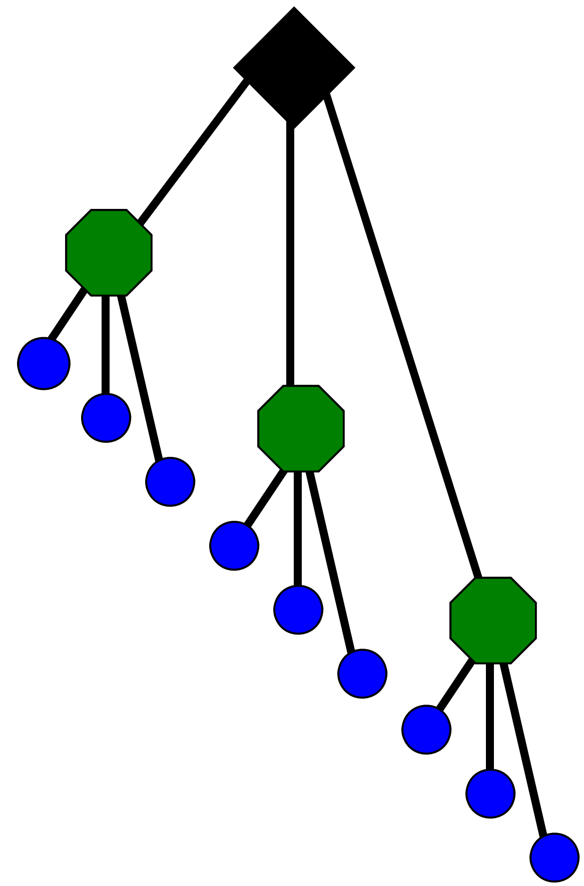


D. Nistér and H. Stewénius, [Scalable Recognition with a Vocabulary Tree](), CVPR 2006



Hierarchical k-means clustering of descriptor space (vocabulary tree)

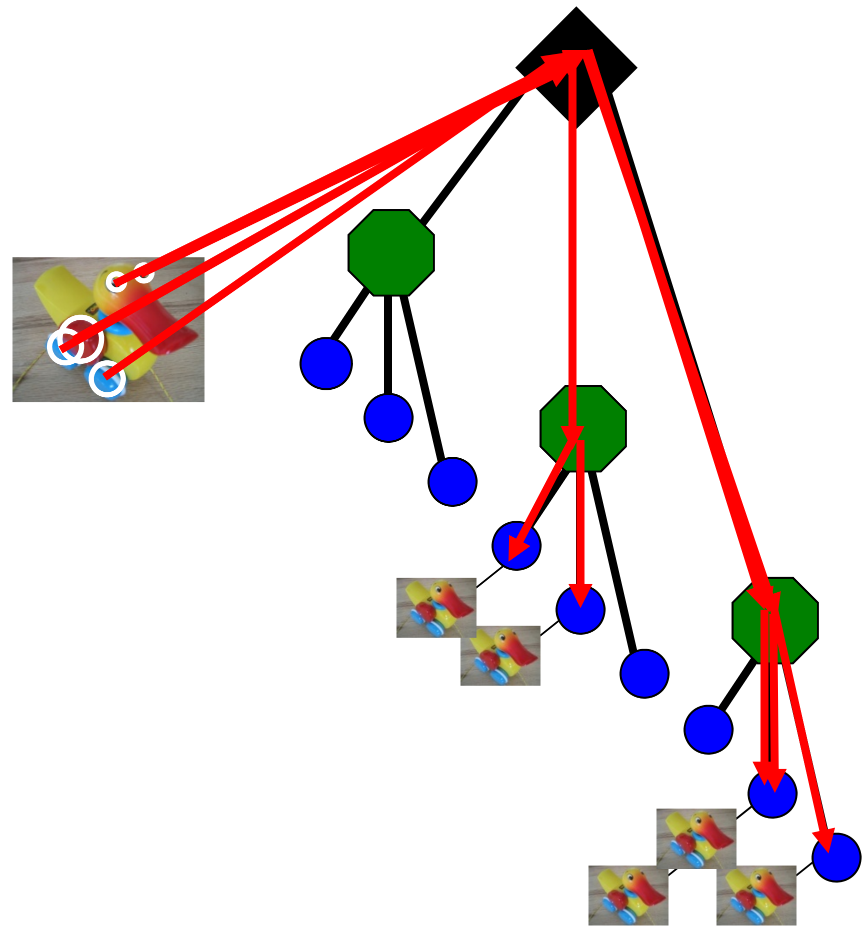
Slide credit: D. Nister



Vocabulary tree/inverted index

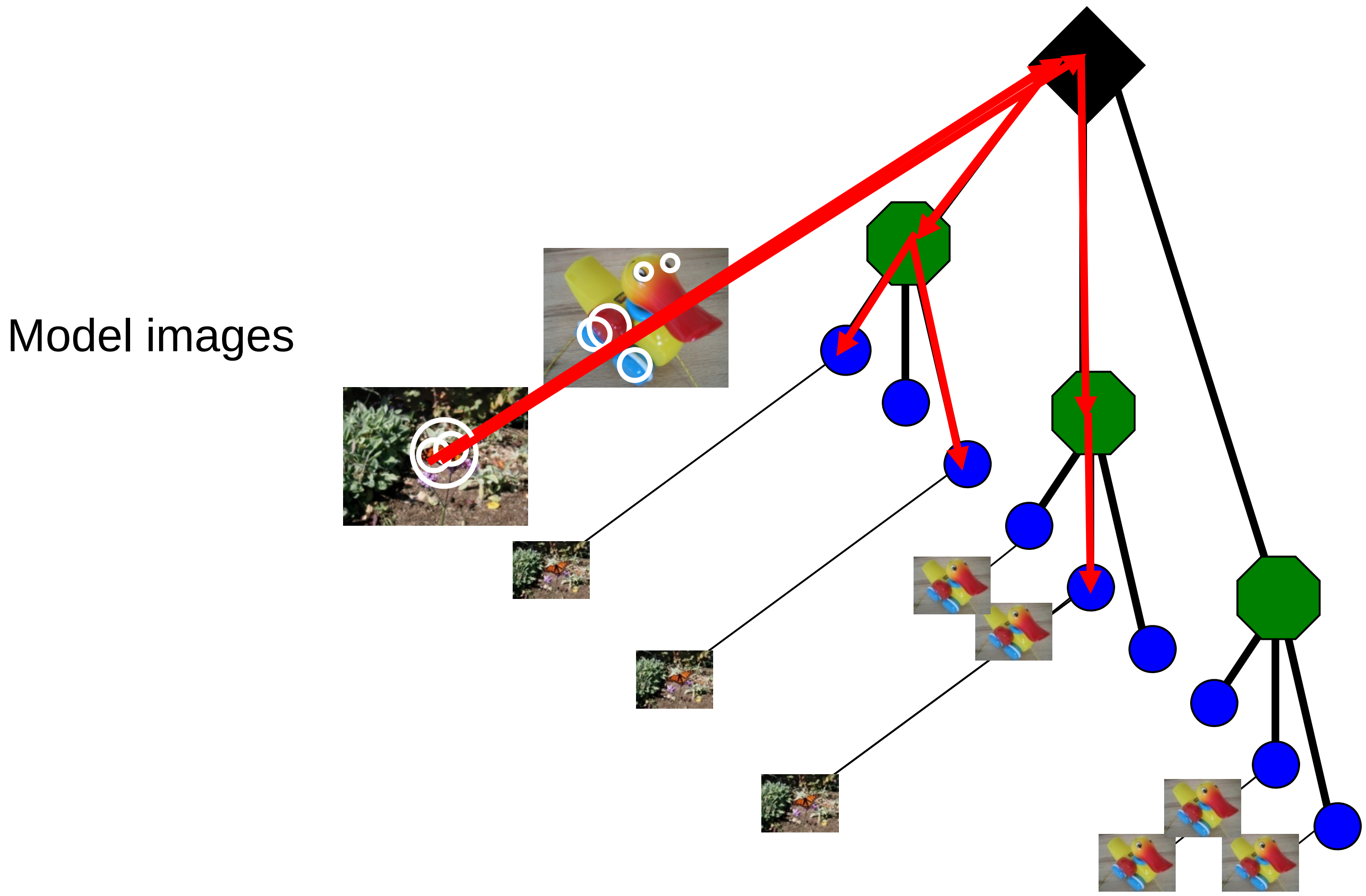
Slide credit: D. Nister

Model images



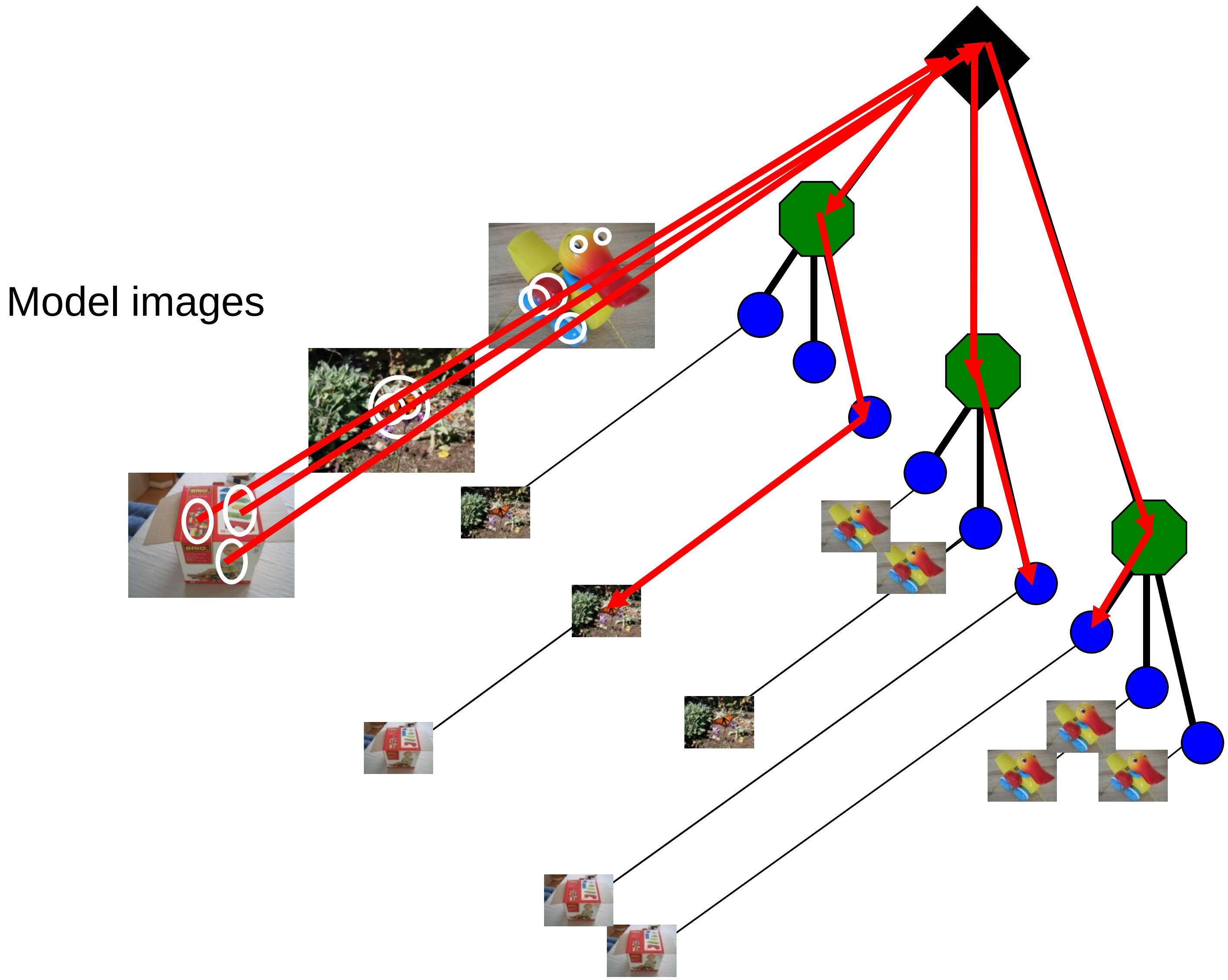
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



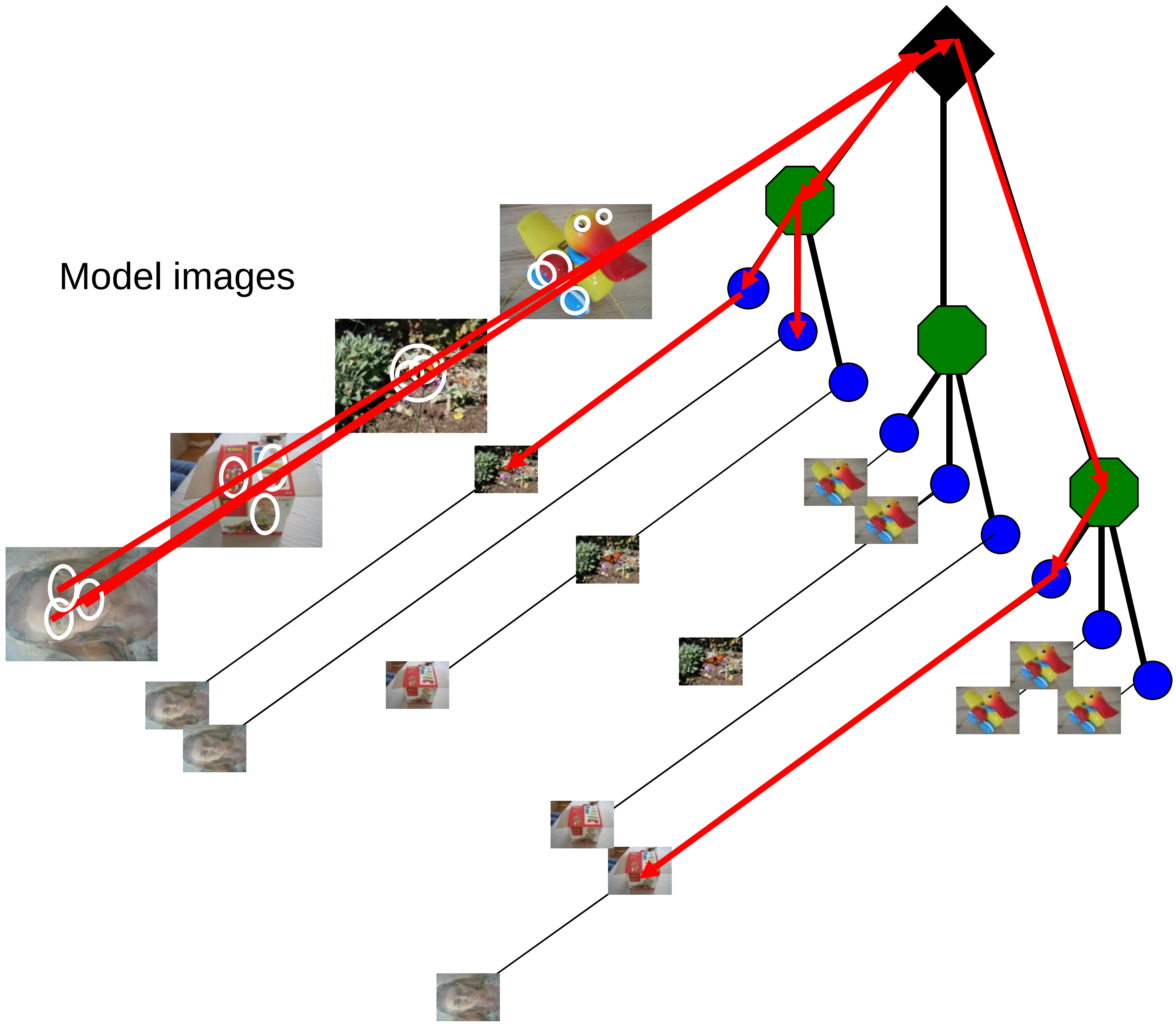
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



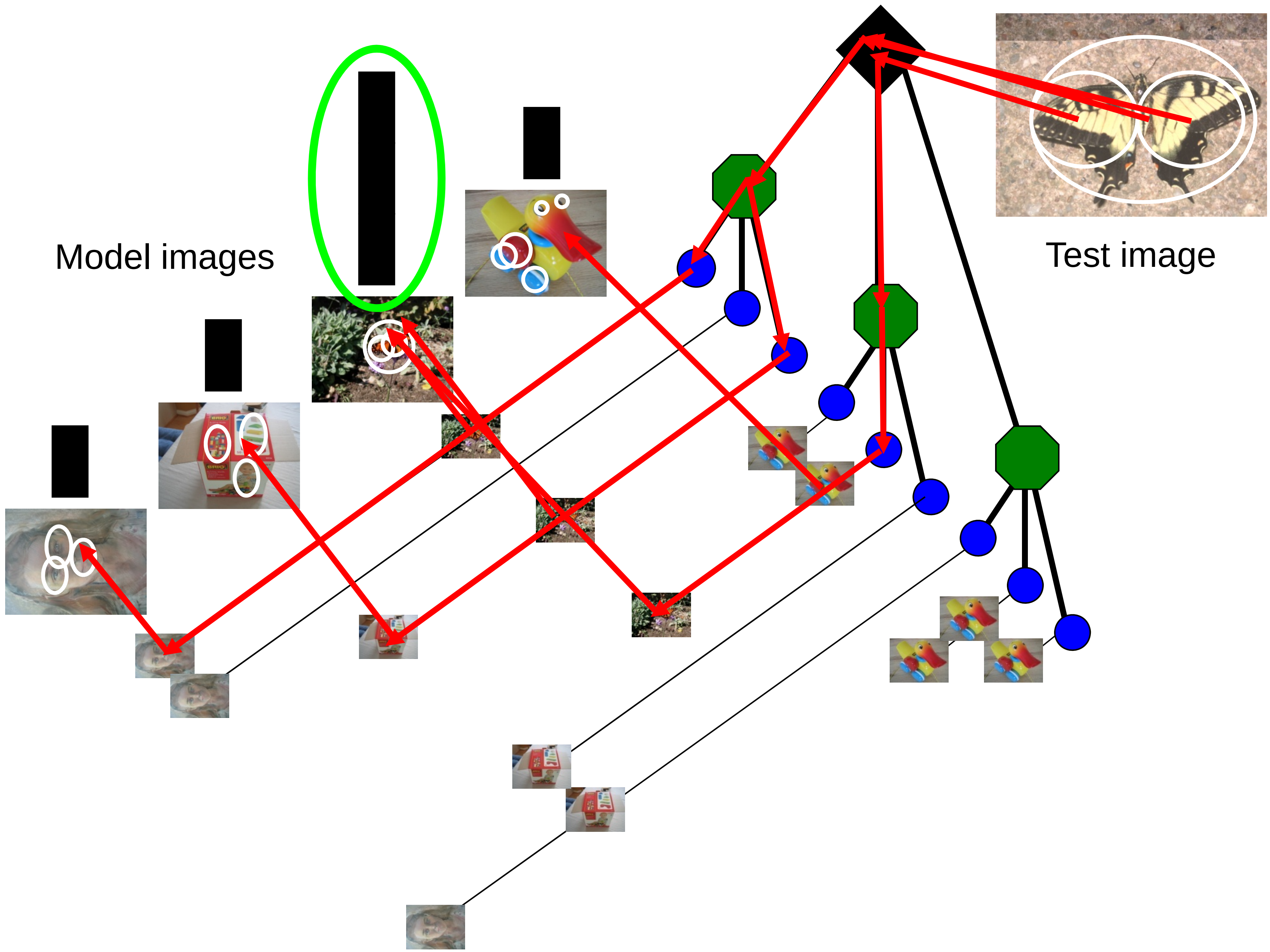
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



Populating the vocabulary tree/inverted index

Slide credit: D. Nister

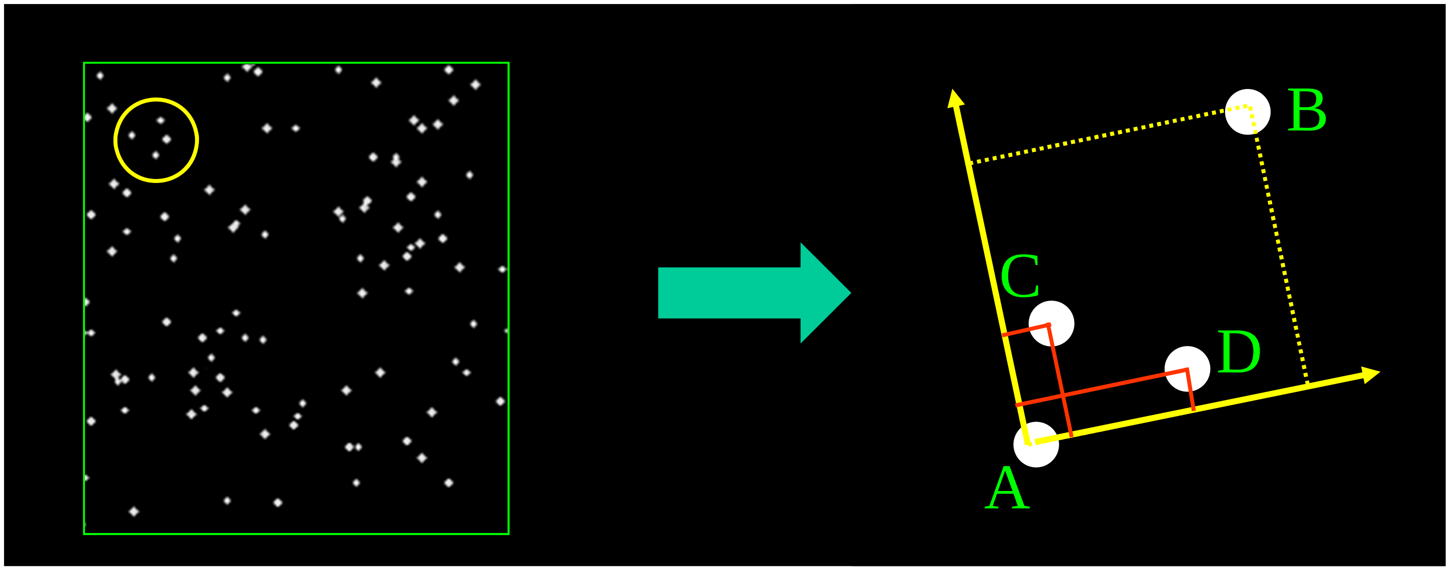


Looking up a test image

Slide credit: D. Nister

Cool application of large-scale alignment: searching the night sky

http://www.astrometry.net/



Slide Credits

Svetlana Lazebnik – UIUC

Derek Hoiem – UIUC

David Forsyth - UIUC

Questions

Image Alignment

Readings for today:

- Forsyth and Ponce chapter 12