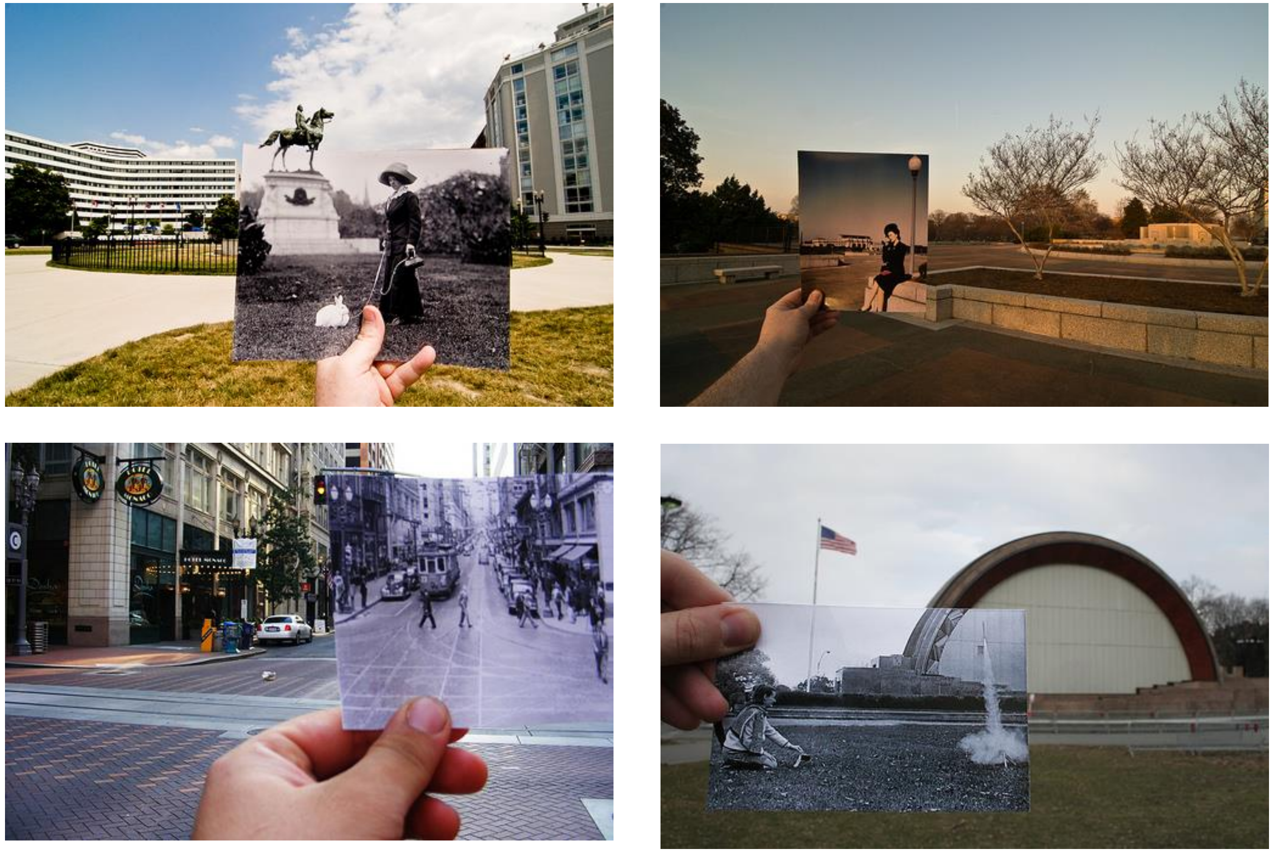
Image Alignment

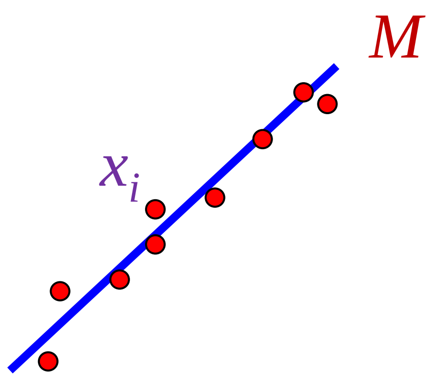
Slides are from Svetlana Lazebnik



Source

Alignment: Fitting of transformations

- Previously: fitting a model to features in one image

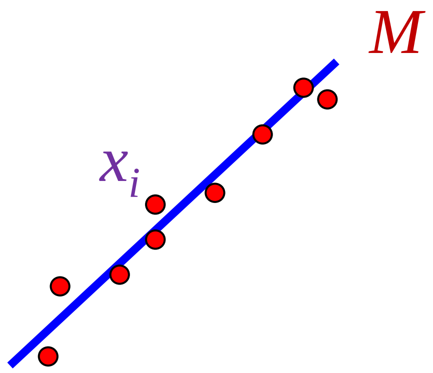


- Given: points $x_1, \ldots, x_n$
- Find: model M that minimizes

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

Alignment: Fitting of transformations

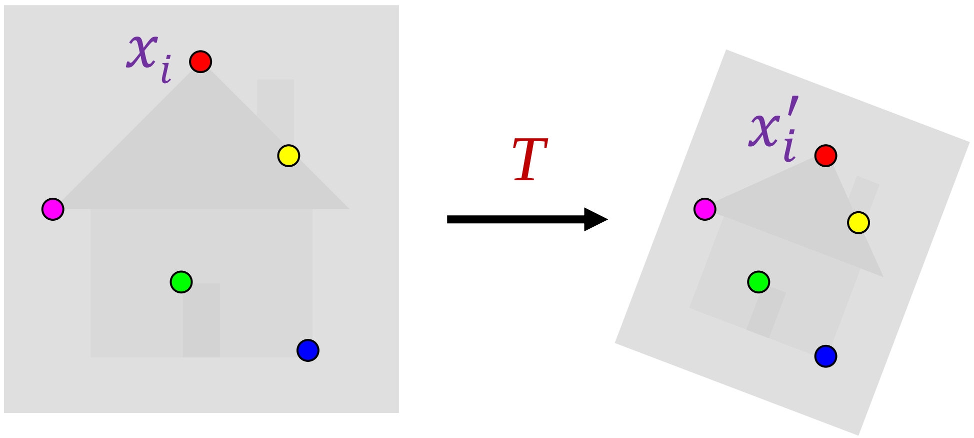
- Previously: fitting a model to features in one image



- Given: points $x_1, \ldots, x_n$
- 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



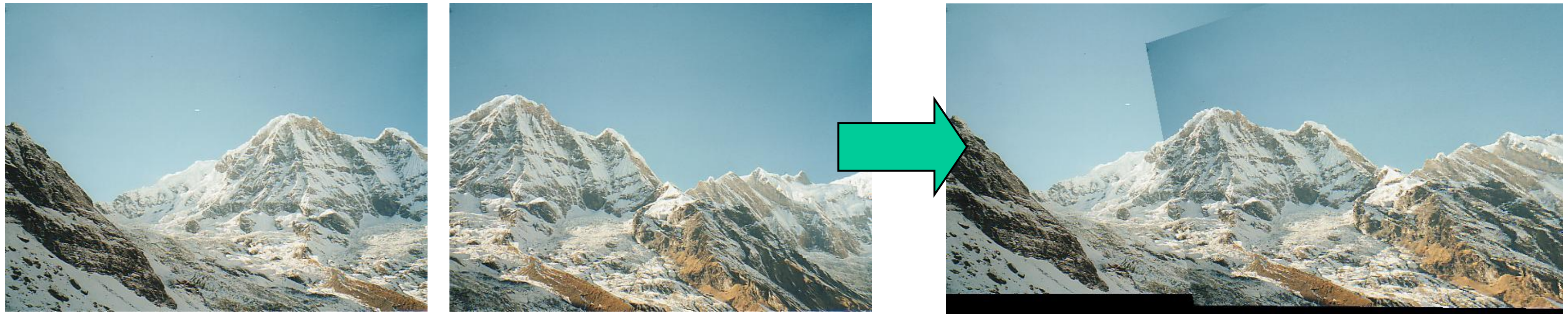
- Given: matches $(x_1, x_1'), \ldots, (x_n, x_n')$
- Find: transformation T that minimizes

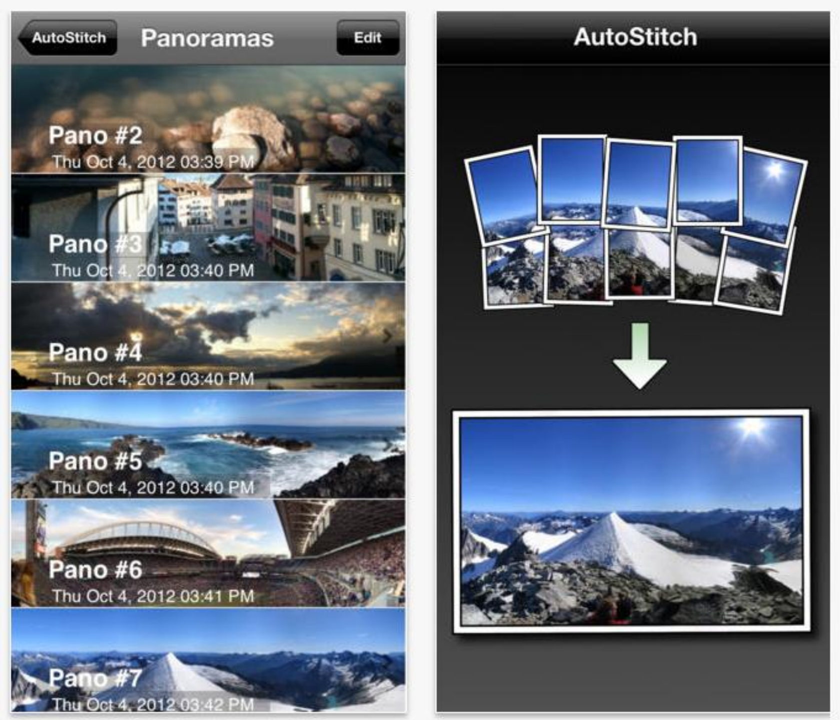
$$\sum_i \text{residual}(T(x_i), x_i')$$

Alignment: Overview

- Motivation
- Fitting of transformations
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC
- Large-scale alignment
  - Inverted indexing
  - Vocabulary trees

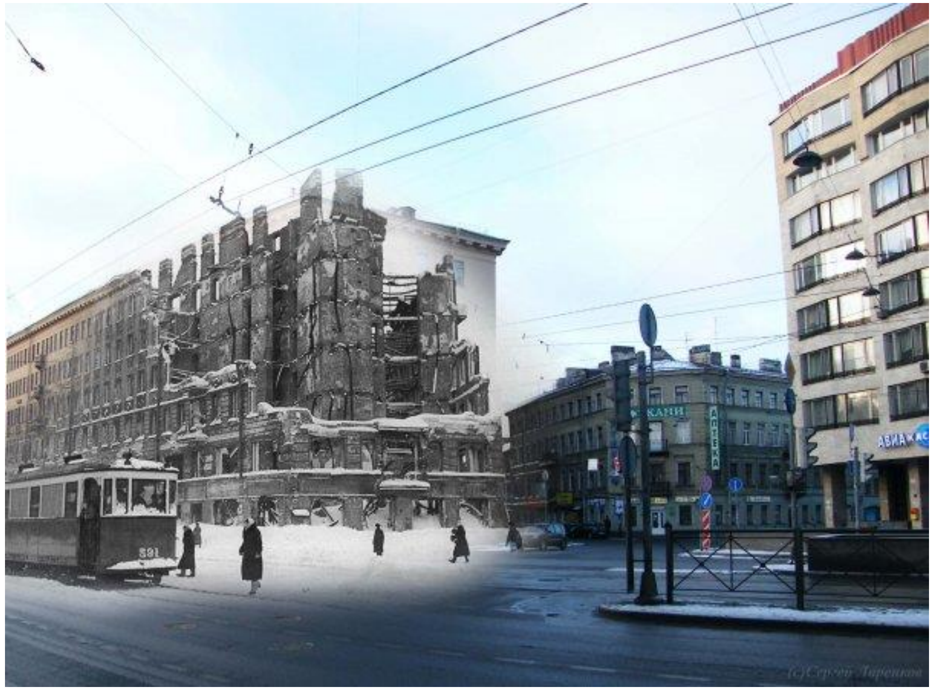
Alignment applications: Panorama Stitching





http://matthewalunbrown.com/autostitch/autostitch.html

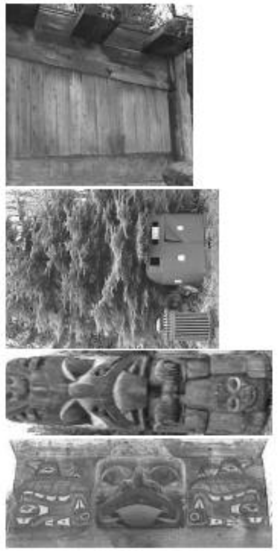
Aligning Contemporary and Historic Images



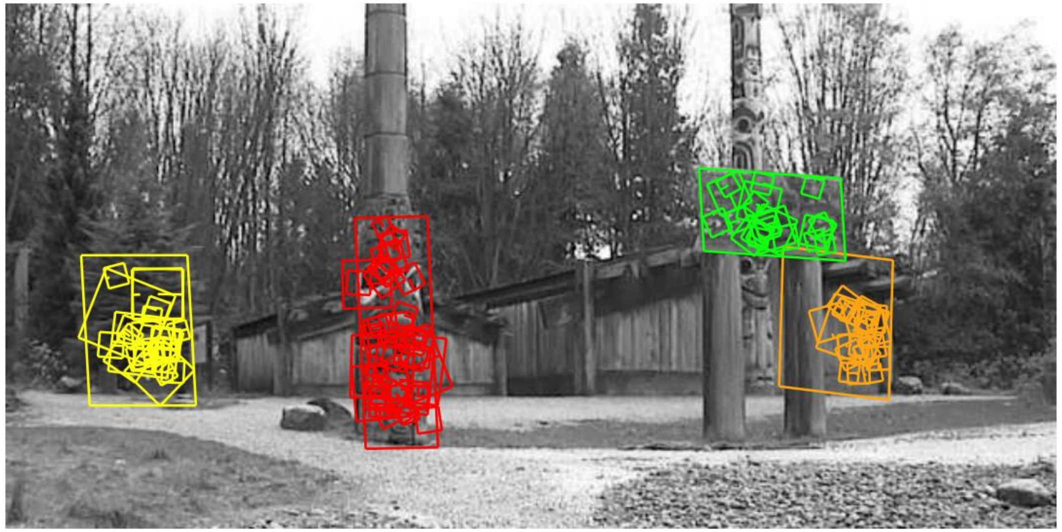
Source

Alignment applications: Instance recognition

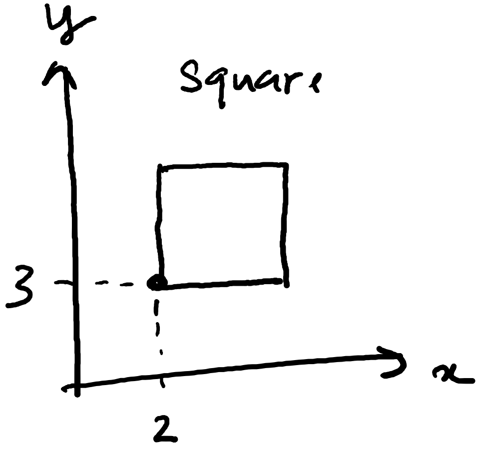
Model images

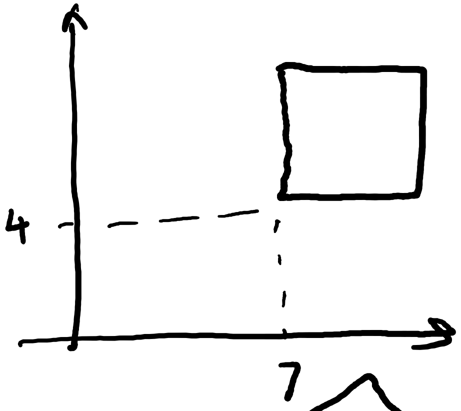


Test image



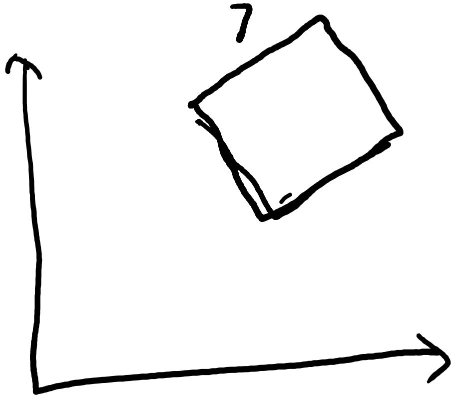
David G. Lowe. [Distinctive image features from scale-invariant keypoints](). *IJCV* 60 (2), pp. 91-110, 2004





Translations

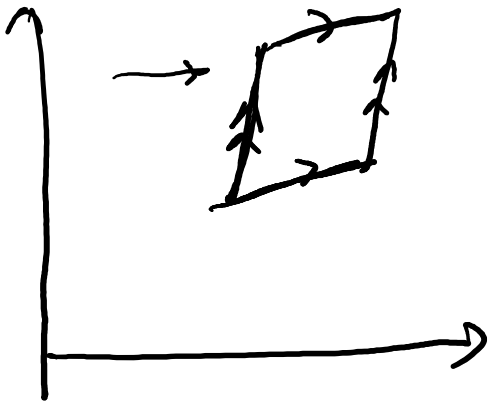
$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}_{3\times 3}$$



Translation + rotation

Euclidean

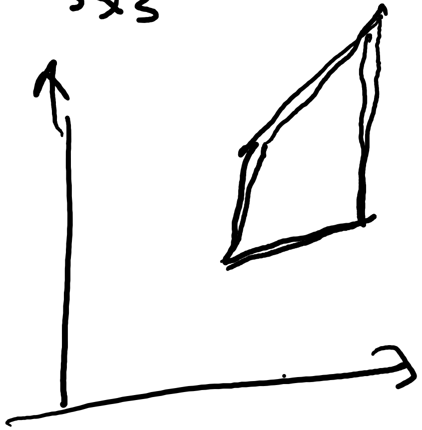
Rigid-body



Translation +

Rotation

Shear



Perspectivity

Homography

Alignment applications: Instance recognition

“identify photographed buildings or objects in query photos and to provide the user with relevant information on them”[1]

[1] T. Weyand and B. Leibe, [Visual landmark recognition from Internet photo collections: A large-scale evaluation](), CVIU 2015

Alignment applications: Large-scale reconstruction

Colosseum: 2,097 images, 819,242 points

Trevi Fountain: 1,935 images, 1,055,153 points

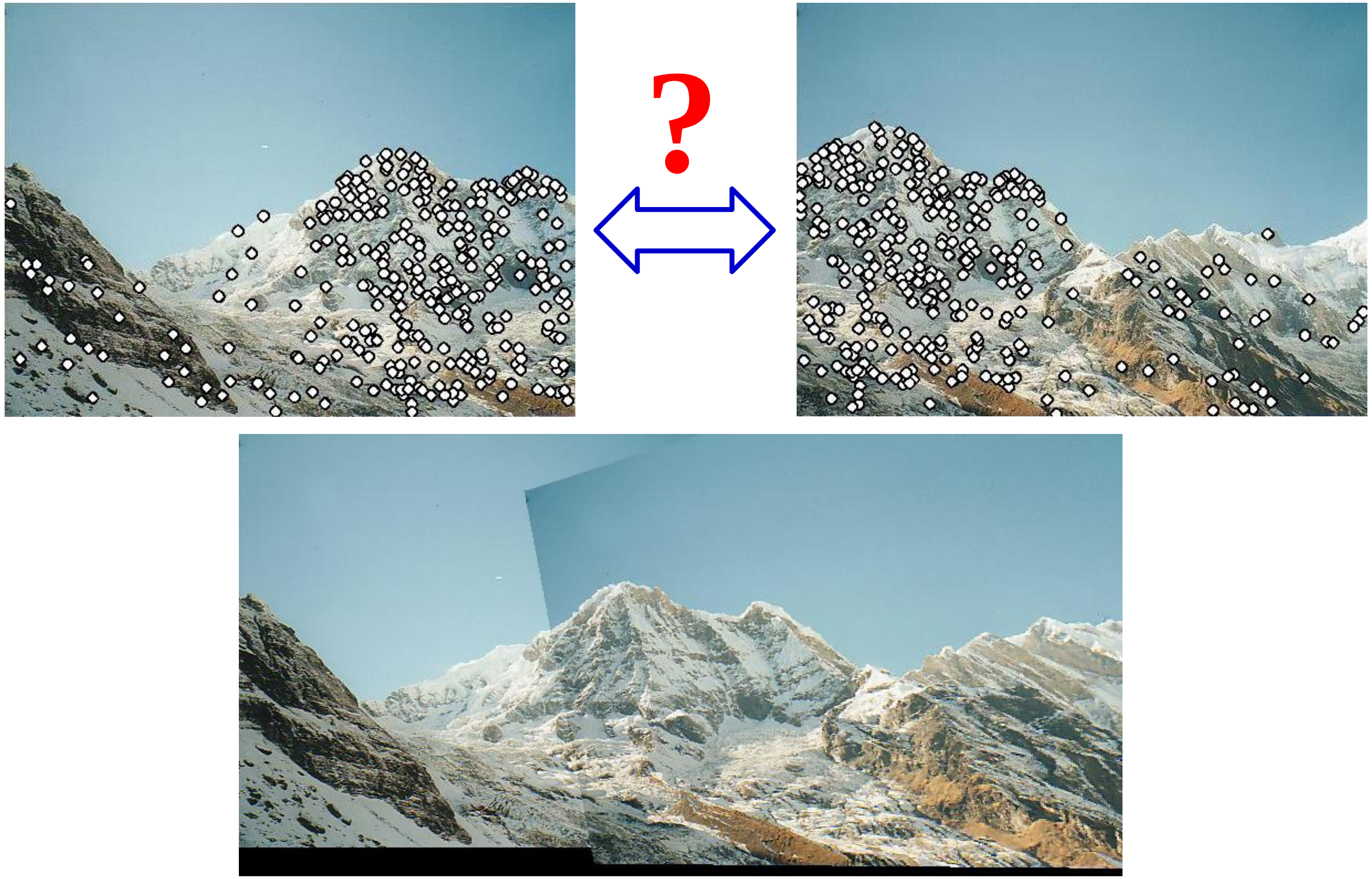
Pantheon: 1,032 images, 530,076 points

Hall of Maps: 275 images, 230,182 points

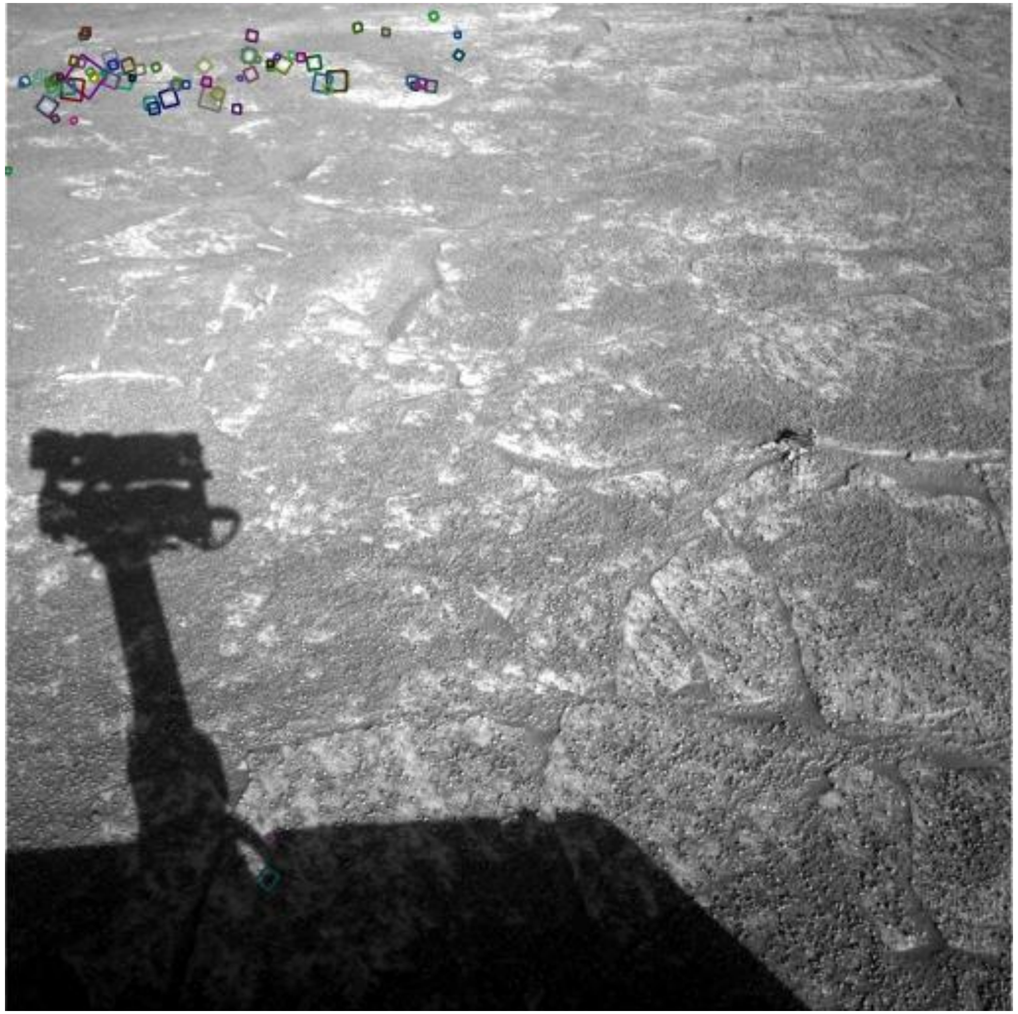
S. Agarwal et al. [Building Rome in a Day](). ICCV 2009

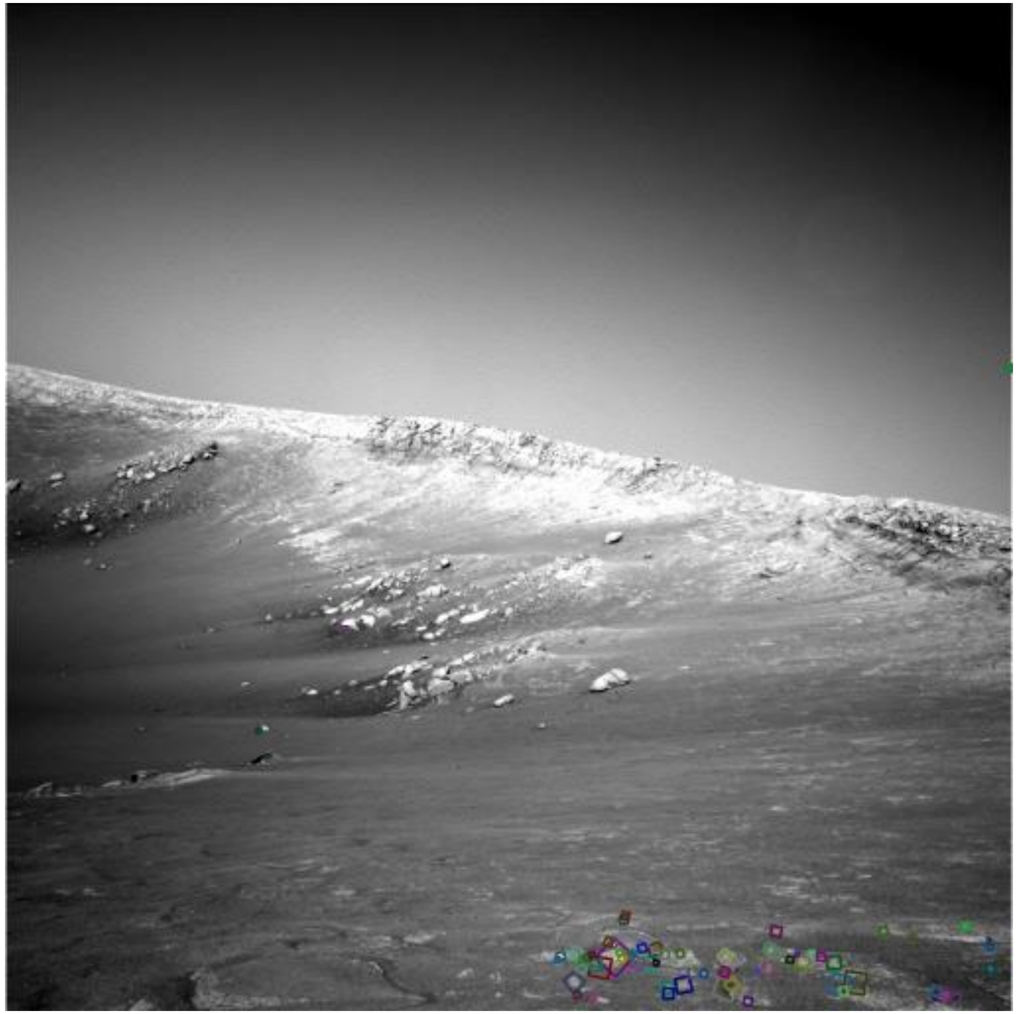
Feature-based alignment

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



Feature-based alignment *really works*!





Source: N. Snavely

Feature-based alignment *really works*!

Source: N. Snavely

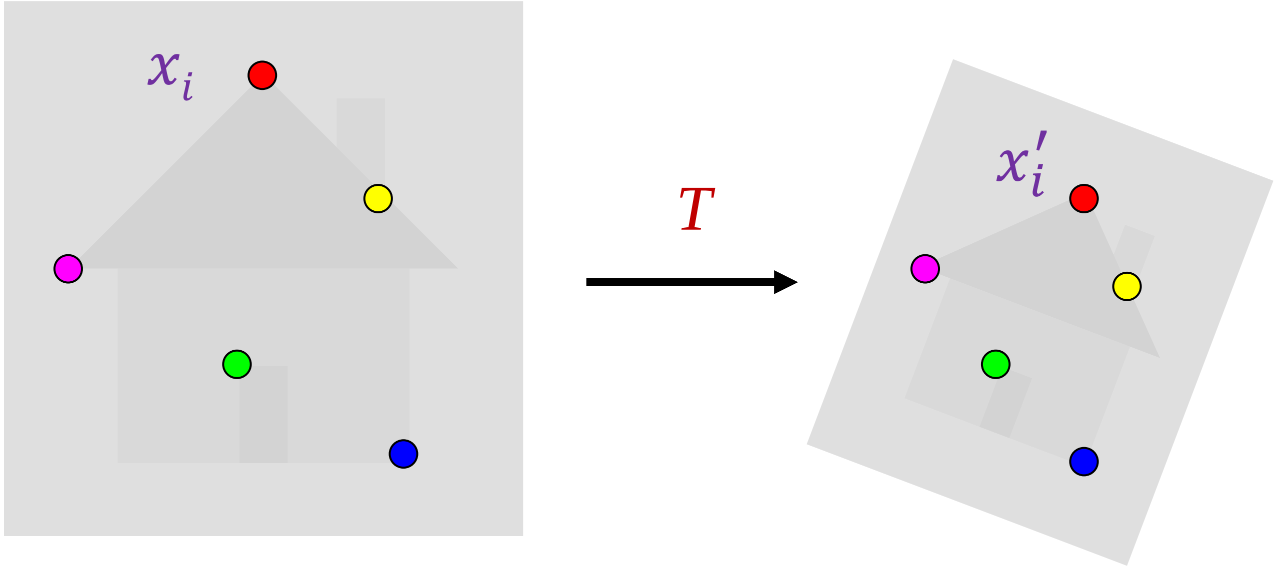
Alignment: Overview

- Motivation
- Fitting of transformations
  - Affine transformations
  - Homographies

Alignment: Fitting of transformations

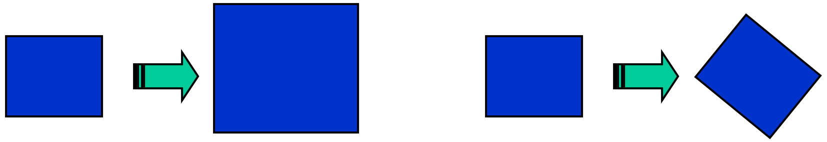
- Given: matches $(x_1, x_1'), \dots, (x_n, x_n')$
- Find: transformation T that minimizes

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

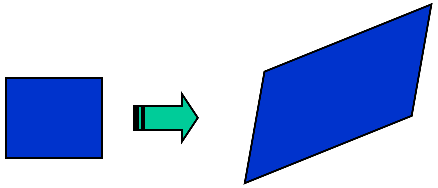


2D transformation models

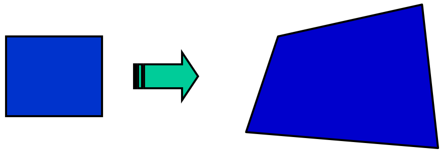
- Similarity (translation, scale, rotation)



- Affine



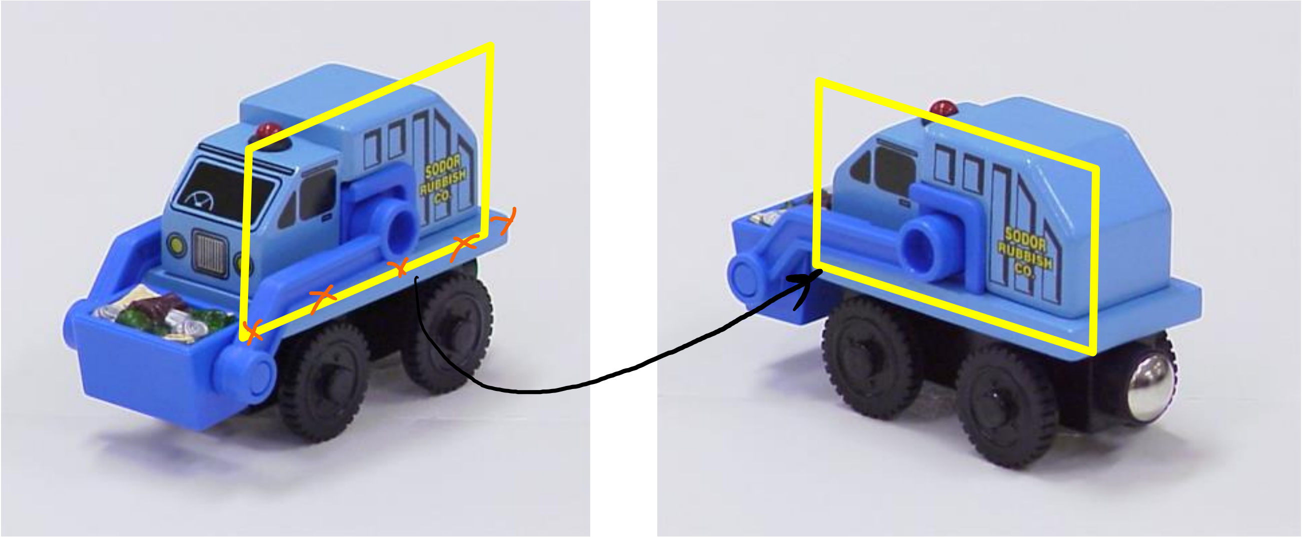
- Projective (homography)



https://en.wikipedia.org/wiki/Affine_transformation

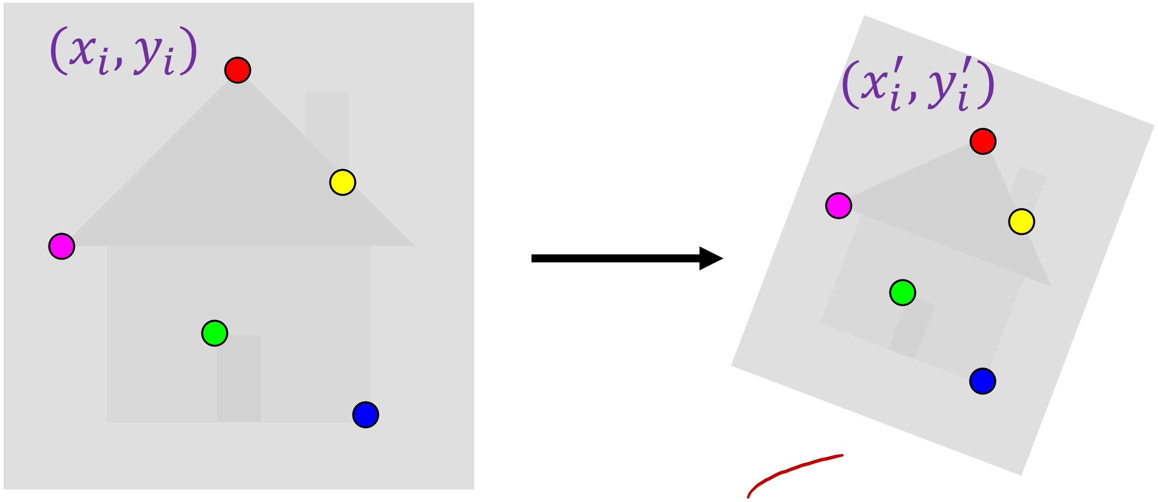
Let's start with affine transformations

- 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?



Transformed points

Original points

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

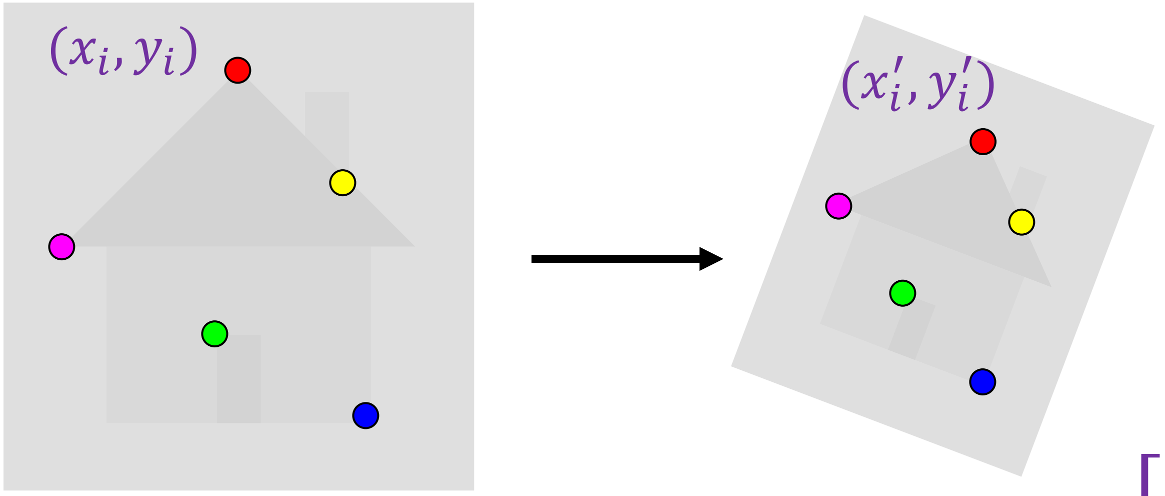
$$\boldsymbol{x}_i' \qquad \boldsymbol{M} \qquad \boldsymbol{x}_i \qquad \boldsymbol{t}$$

Want to find $\boldsymbol{M}$, $\boldsymbol{t}$ to minimize

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

Fitting an affine transformation

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

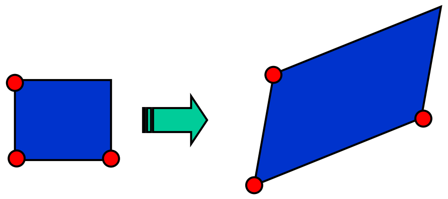


$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

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

Fitting an affine transformation

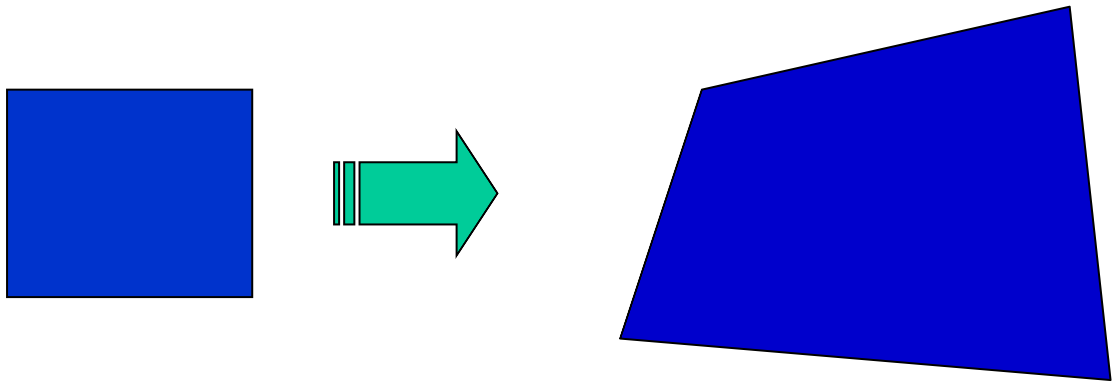
- How many matches do we need to solve for the transformation parameters?



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

Fitting a plane projective transformation

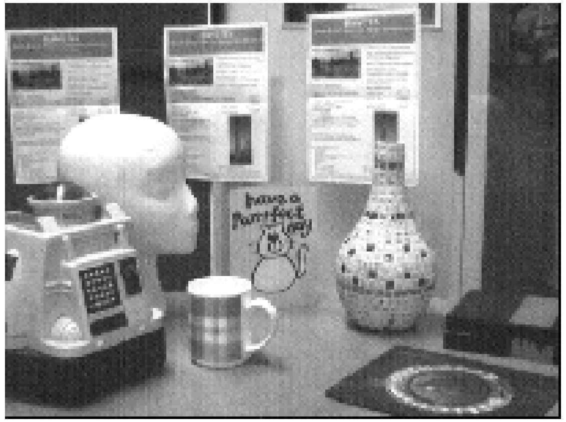
- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)



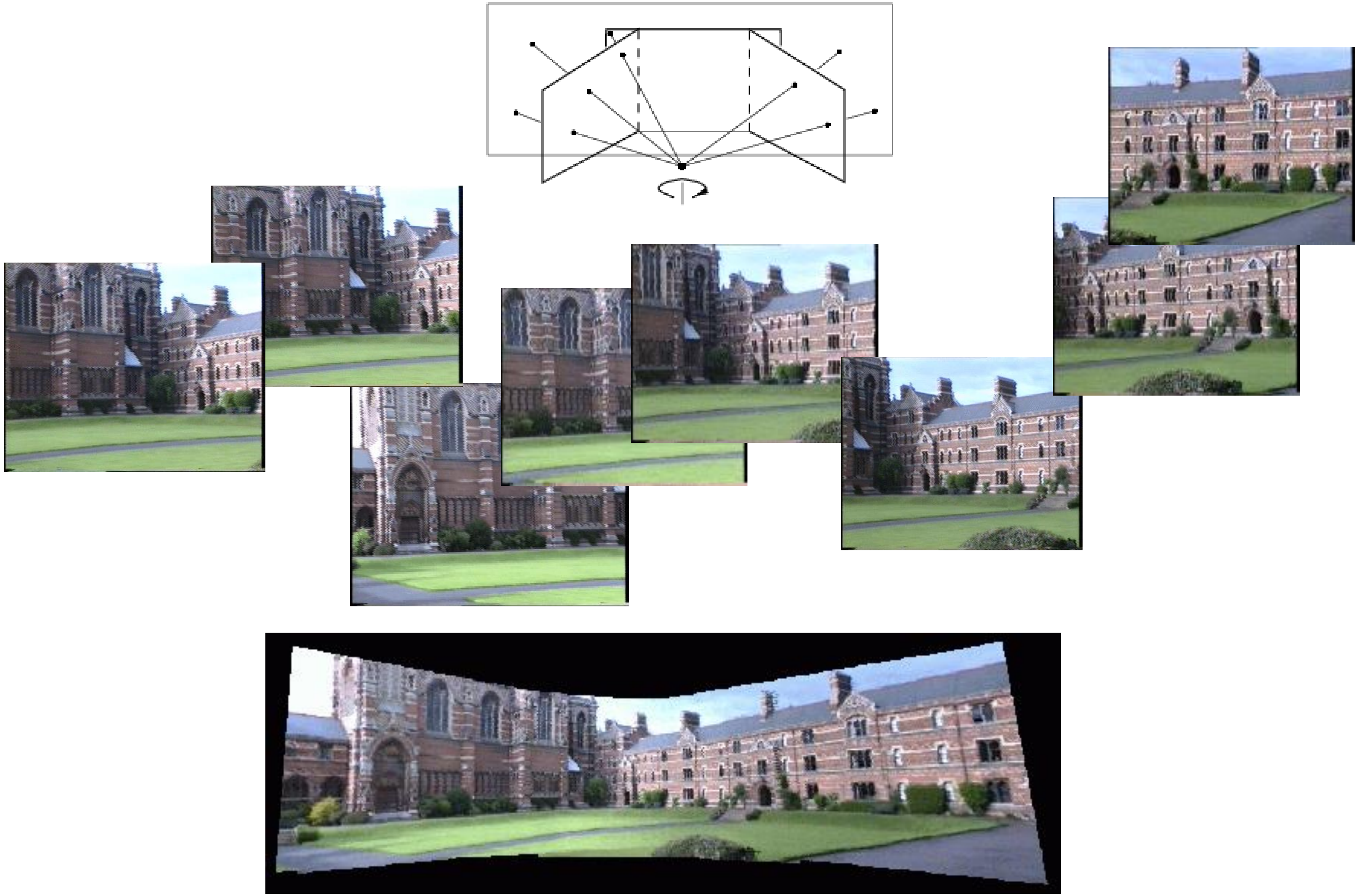
Where do homographies arise in the real world?

- The transformation between two views of a planar surface


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




Application: Panorama stitching

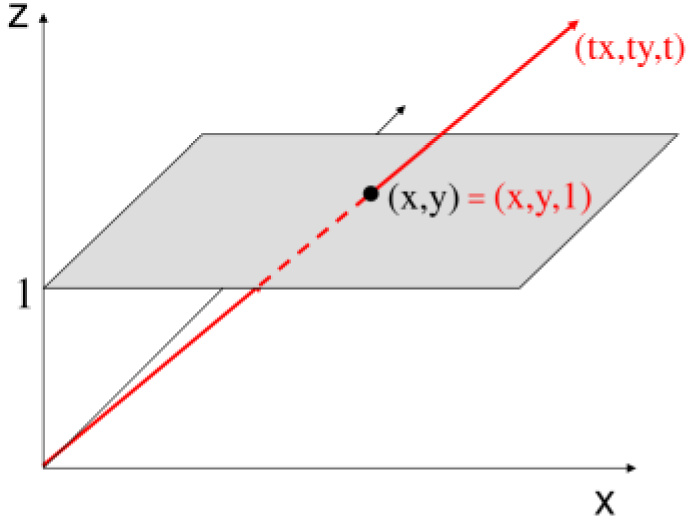


Source: Hartley & Zisserman

Fitting a homography

$$(x, y) \rightarrow (x, y, 1) \equiv (tx, ty, t)$$

$$(X, Y, W) \rightarrow \left(\frac{X}{W}, \frac{Y}{W}\right) = (x, y)$$



- Recall: 2D homogeneous coordinates

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

Converting *to* homogeneous image coordinates

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

Converting *from* homogeneous image coordinates

- Equation for homography in homogeneous coordinates:

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

$$\boxed{\lambda \boldsymbol{x}' = \boldsymbol{H}\boldsymbol{x}}$$

scale factor λ

Fitting a homography

$$H_{3\times 3} = \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

- Constraint from a match $(\boldsymbol{x}_i, \boldsymbol{x}_i')$: $\lambda \boldsymbol{x}_i' = \boldsymbol{H}\boldsymbol{x}_i$
- How can we get rid of the scale factor λ?
- Cross product trick: $\boldsymbol{x}_i' \times \boldsymbol{H}\boldsymbol{x}_i = \boldsymbol{0}$
- Recall cross product:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \times \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} bc' - b'c \\ ca' - c'a \\ ab' - a'b \end{pmatrix}$$

- Let $\boldsymbol{h}_1^T, \boldsymbol{h}_2^T, \boldsymbol{h}_3^T$ be the rows of $\boldsymbol{H}$. Then

$$\boldsymbol{x}_i' \times \underline{\boldsymbol{H}\boldsymbol{x}_i} = \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} \times \begin{pmatrix} \boldsymbol{h}_1^T\boldsymbol{x}_i \\ \boldsymbol{h}_2^T\boldsymbol{x}_i \\ \boldsymbol{h}_3^T\boldsymbol{x}_i \end{pmatrix} = \underline{\begin{pmatrix} y_i'\boldsymbol{h}_3^T\boldsymbol{x}_i - \boldsymbol{h}_2^T\boldsymbol{x}_i \\ \boldsymbol{h}_1^T\boldsymbol{x}_i - x_i'\boldsymbol{h}_3^T\boldsymbol{x}_i \\ x_i'\boldsymbol{h}_2^T\boldsymbol{x}_i - y_i'\boldsymbol{h}_1^T\boldsymbol{x}_i \end{pmatrix}}$$

Fitting a homography

- Constraint from a match $(\boldsymbol{x}_i, \boldsymbol{x}_i')$:

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

- Rearranging the terms:

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

- Are these equations independent?

Fitting a homography

- Final linear system:

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

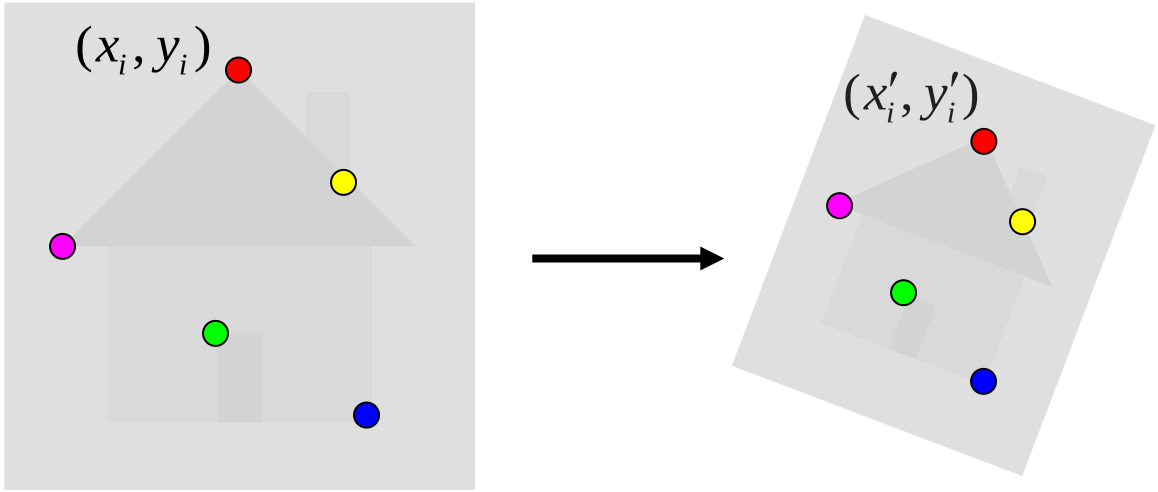
- Homogeneous least squares: find $\boldsymbol{h}$ minimizing $\|A\boldsymbol{h}\|^2$
  - Solution is eigenvector of A^TA corresponding to smallest eigenvalue
- What is the minimum number of matches needed for a solution?
  - Four: A has 8 degrees of freedom (9 parameters, but scale is arbitrary), one match gives us two linearly independent equations

Alignment: Overview

- Motivation
- Fitting of transformations
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC

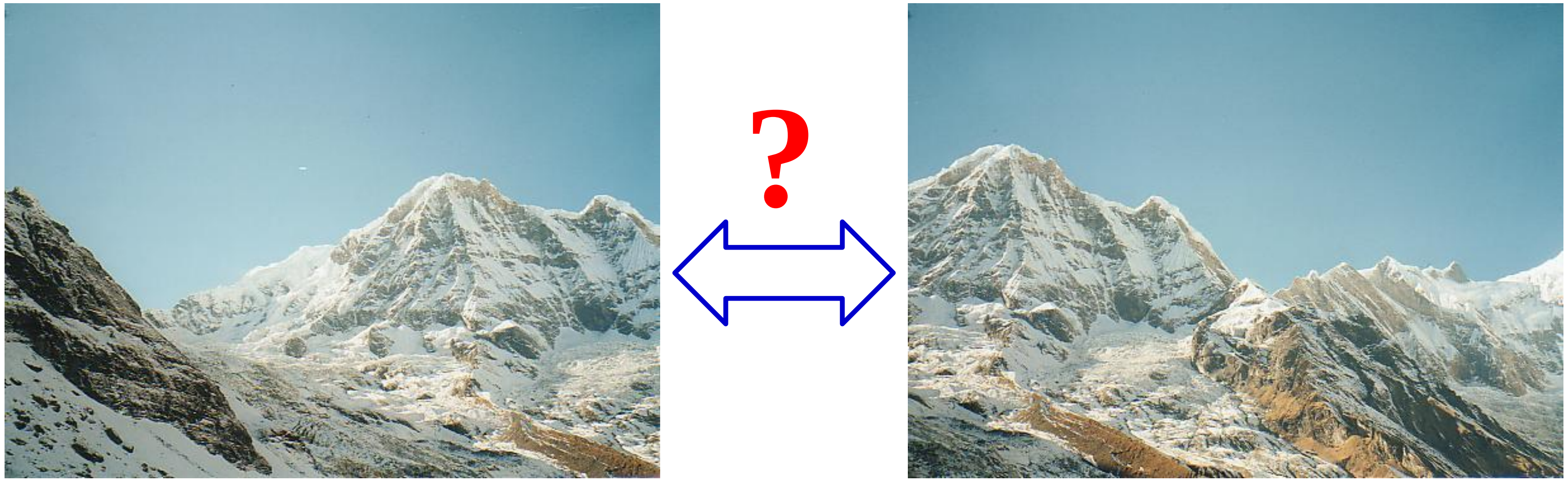
Robust feature-based alignment

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



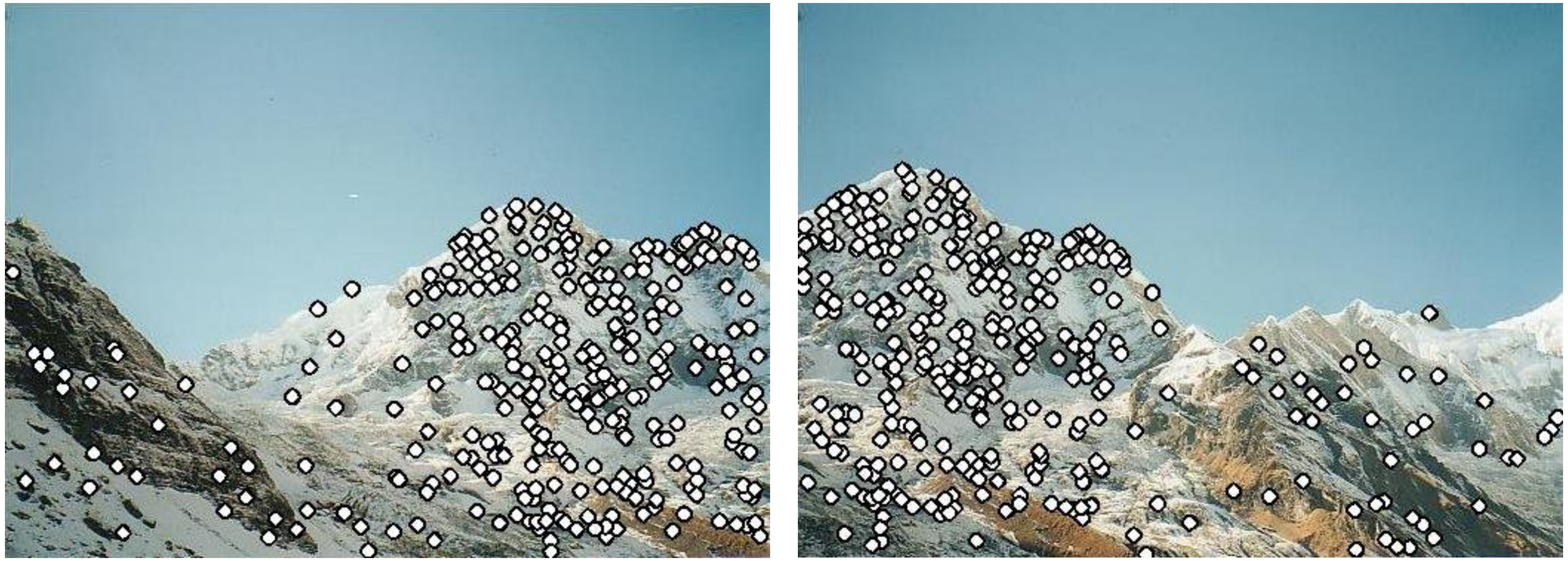
Robust feature-based alignment

- So far, we've assumed that we are given a set of 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

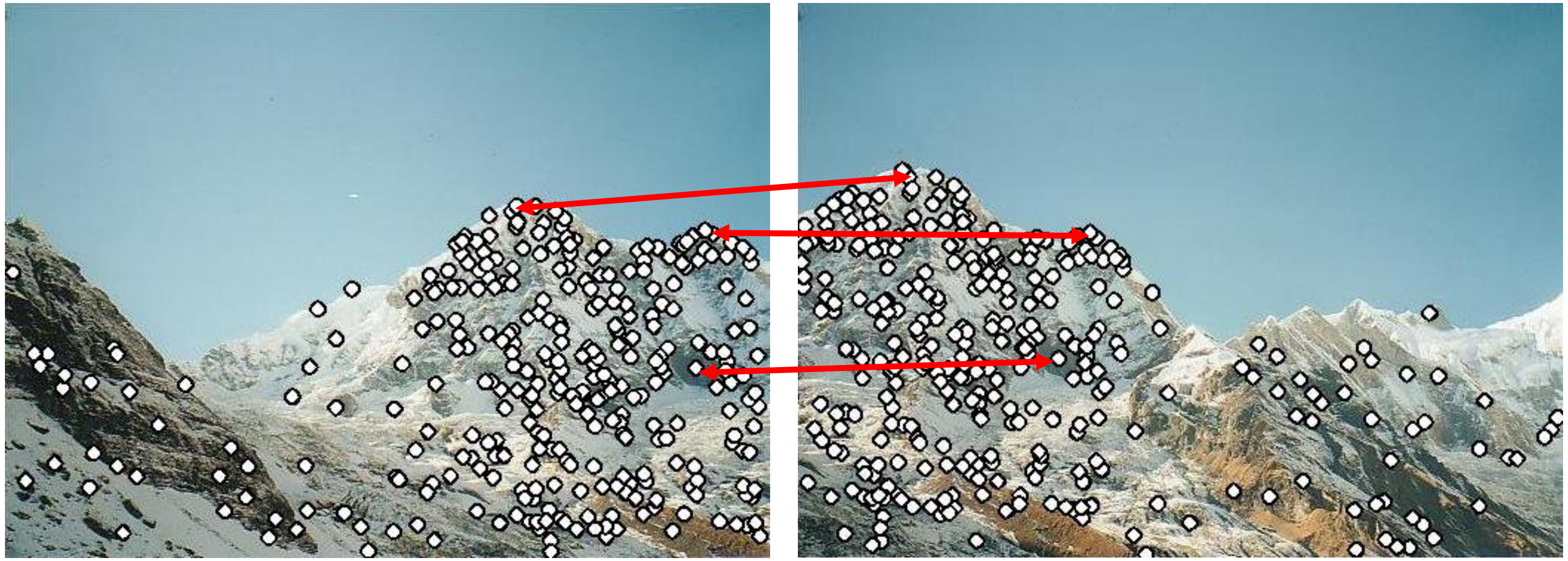


- Extract features

Robust feature-based alignment

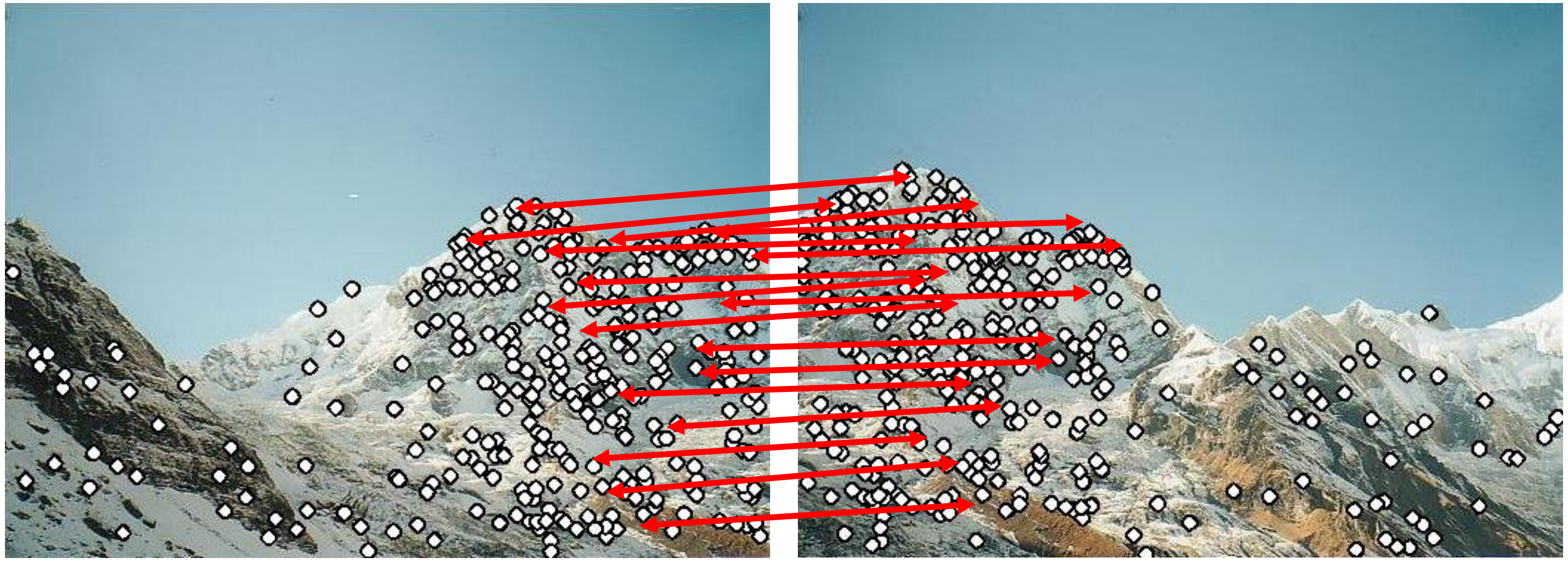
- Extract features
- Compute *putative matches*

Robust feature-based alignment



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

Robust feature-based alignment

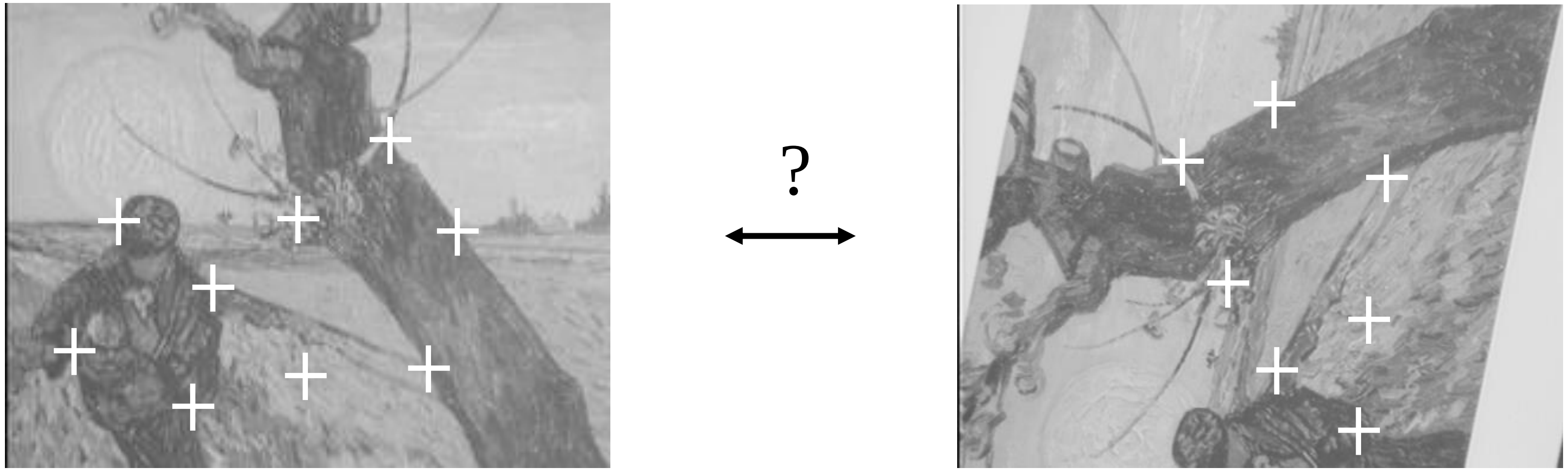


- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation T
  - *Verify* transformation (search for other matches consistent with T)

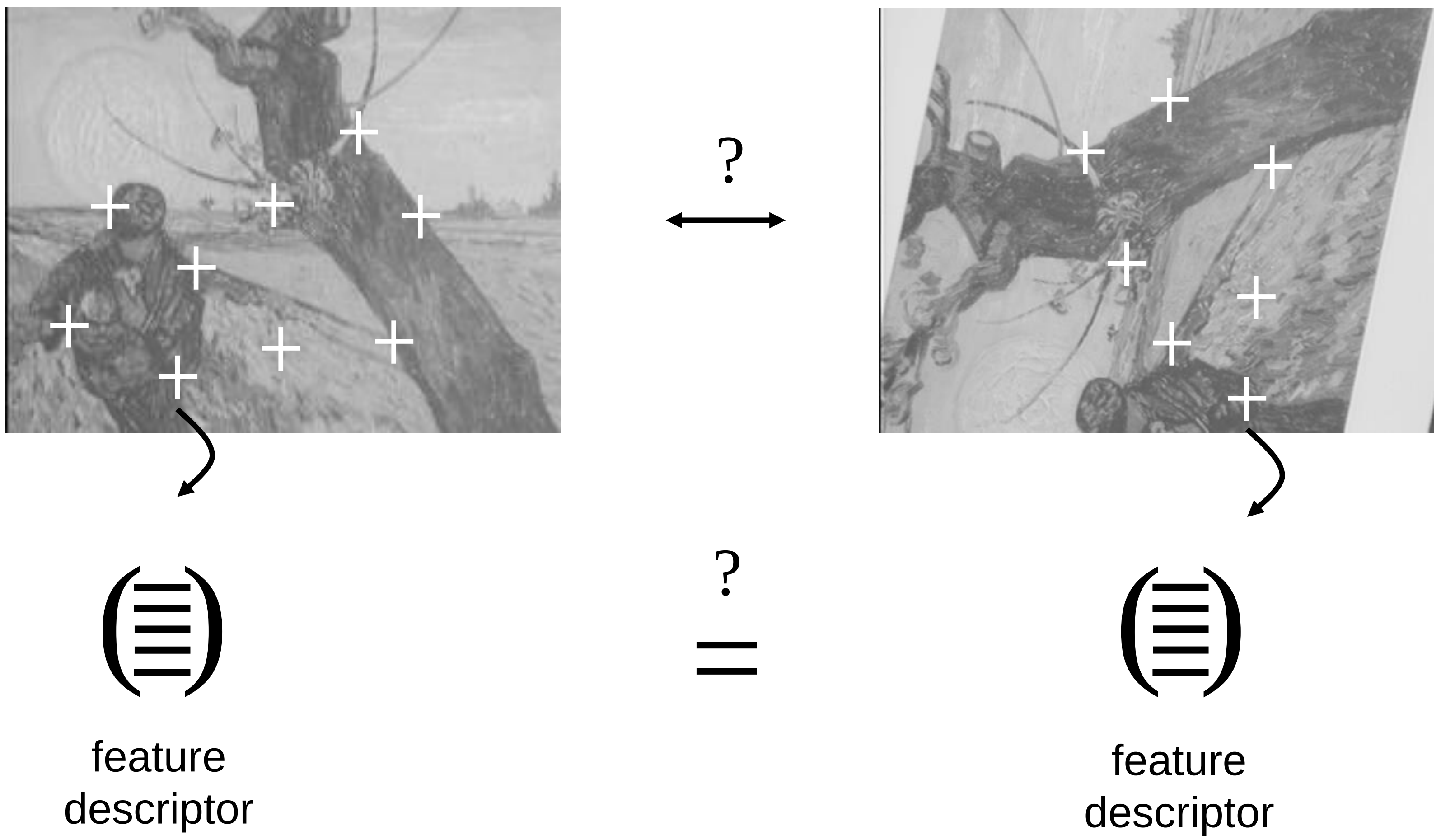
Robust feature-based alignment

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

Generating putative correspondences



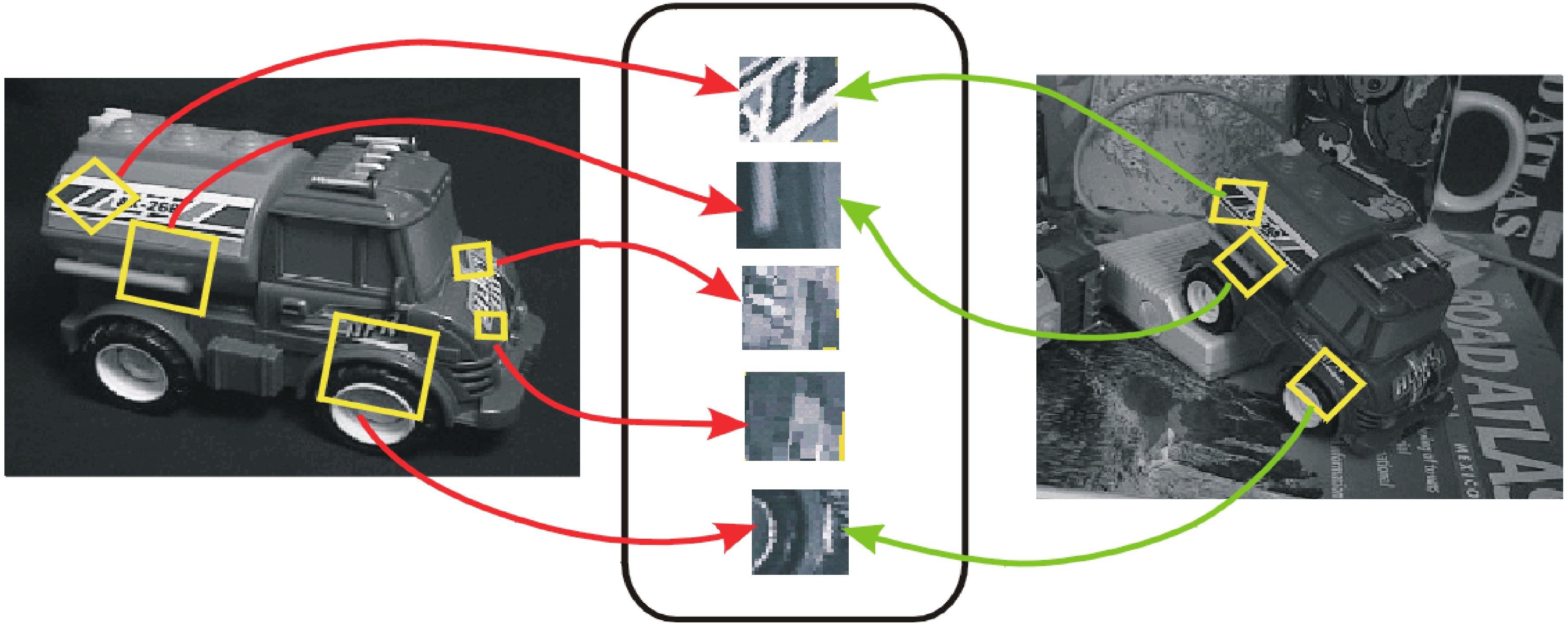
Generating putative correspondences



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

Feature descriptors

- Recall: feature detection vs. feature description



Comparing feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors $\boldsymbol{u}$ and $\boldsymbol{v}$?
  - **Sum of squared differences** (SSD):

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

  - **Normalized correlation**: dot product between $\boldsymbol{u}$ and $\boldsymbol{v}$ normalized to have zero mean and unit norm:

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

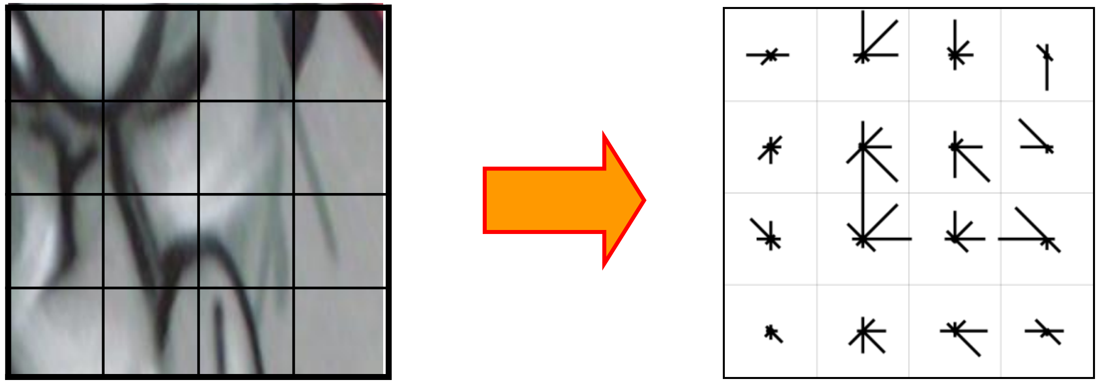
  - Why would we prefer normalized correlation over SSD?

Disadvantage of intensity vectors as descriptors

- Small deformations can affect the matching score a lot

Feature descriptors: Scale-Invariant Feature Transform SIFT

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



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

Feature descriptors: SIFT

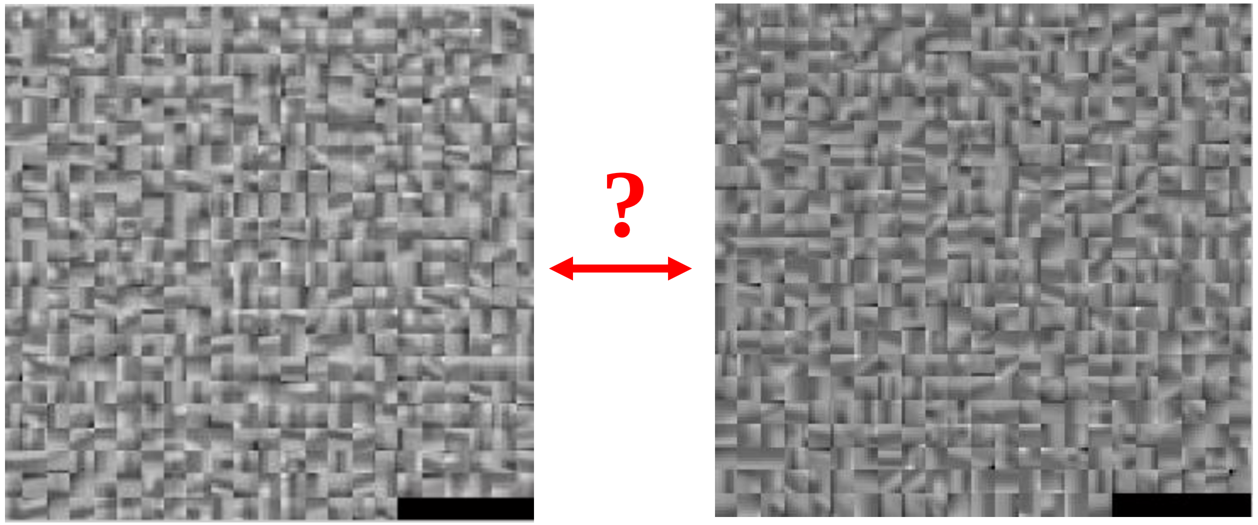
- Descriptor computation:
  - Divide patch into 4×4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions

- What are the advantages of SIFT descriptor over raw pixel values?
  - Gradients are 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.

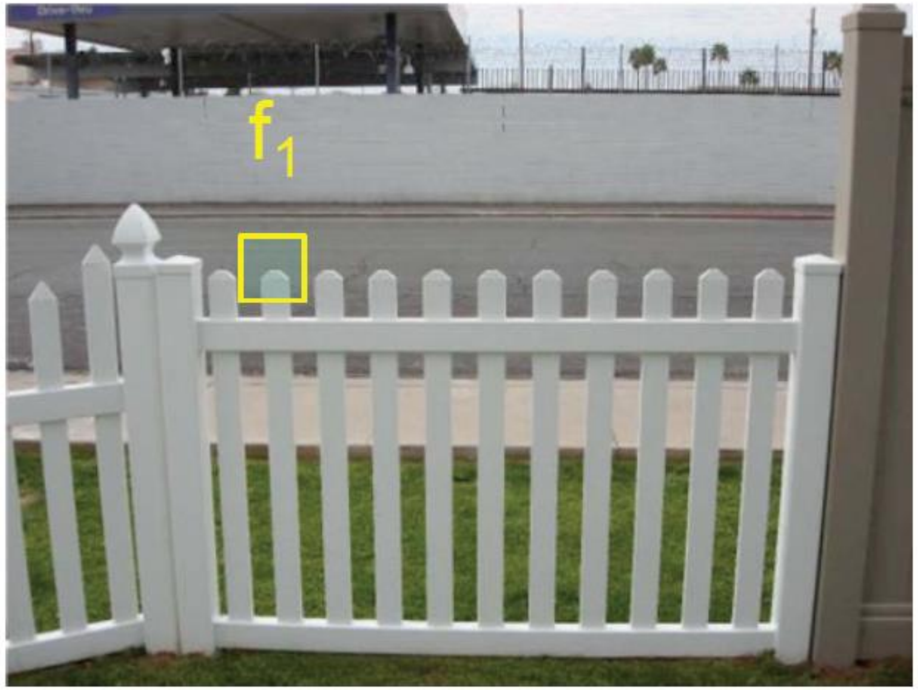
Generating putative correspondences

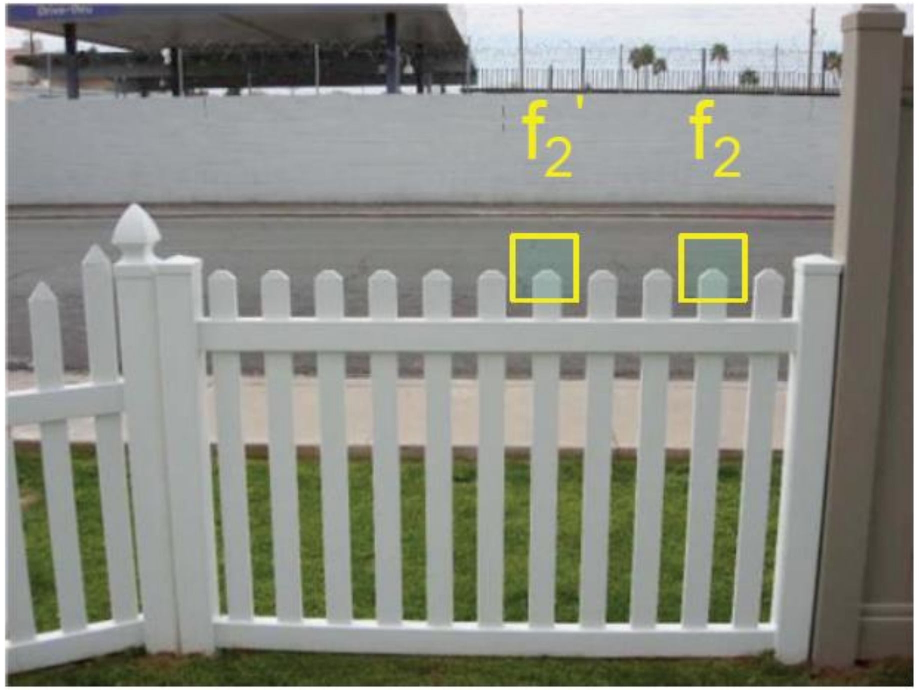
- For each patch in one image, find a short list of patches in the other image that could match it based solely on appearance



Rejection of ambiguous matches

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

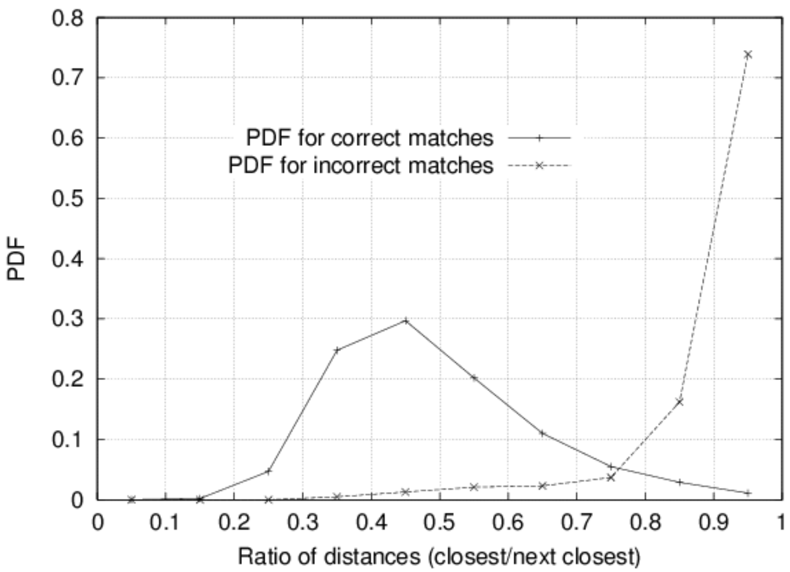




Source: Y. Furukawa

Rejection of ambiguous 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 found to provide good separation

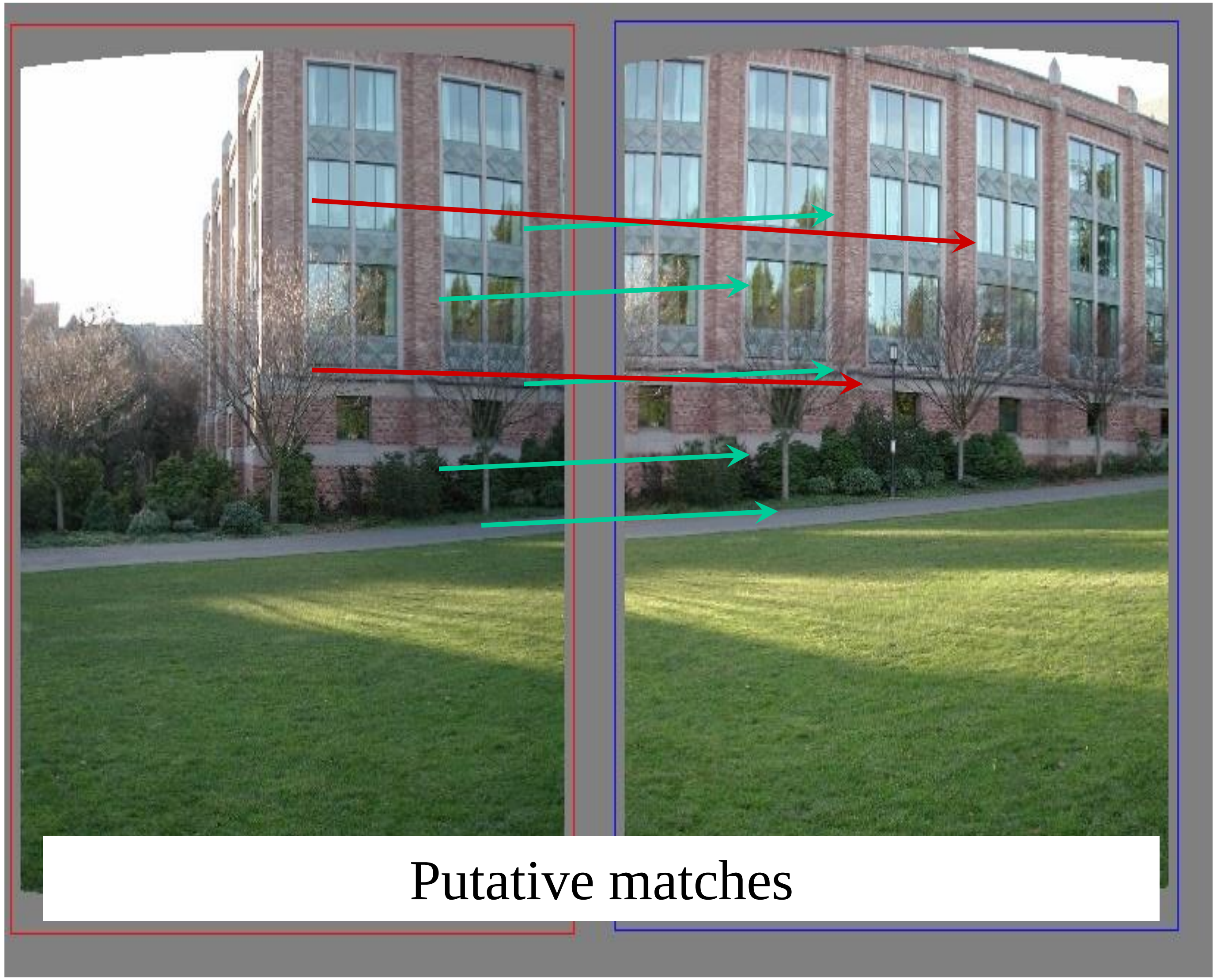
David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV* 60 (2), pp. 91-110, 2004.

Robust alignment

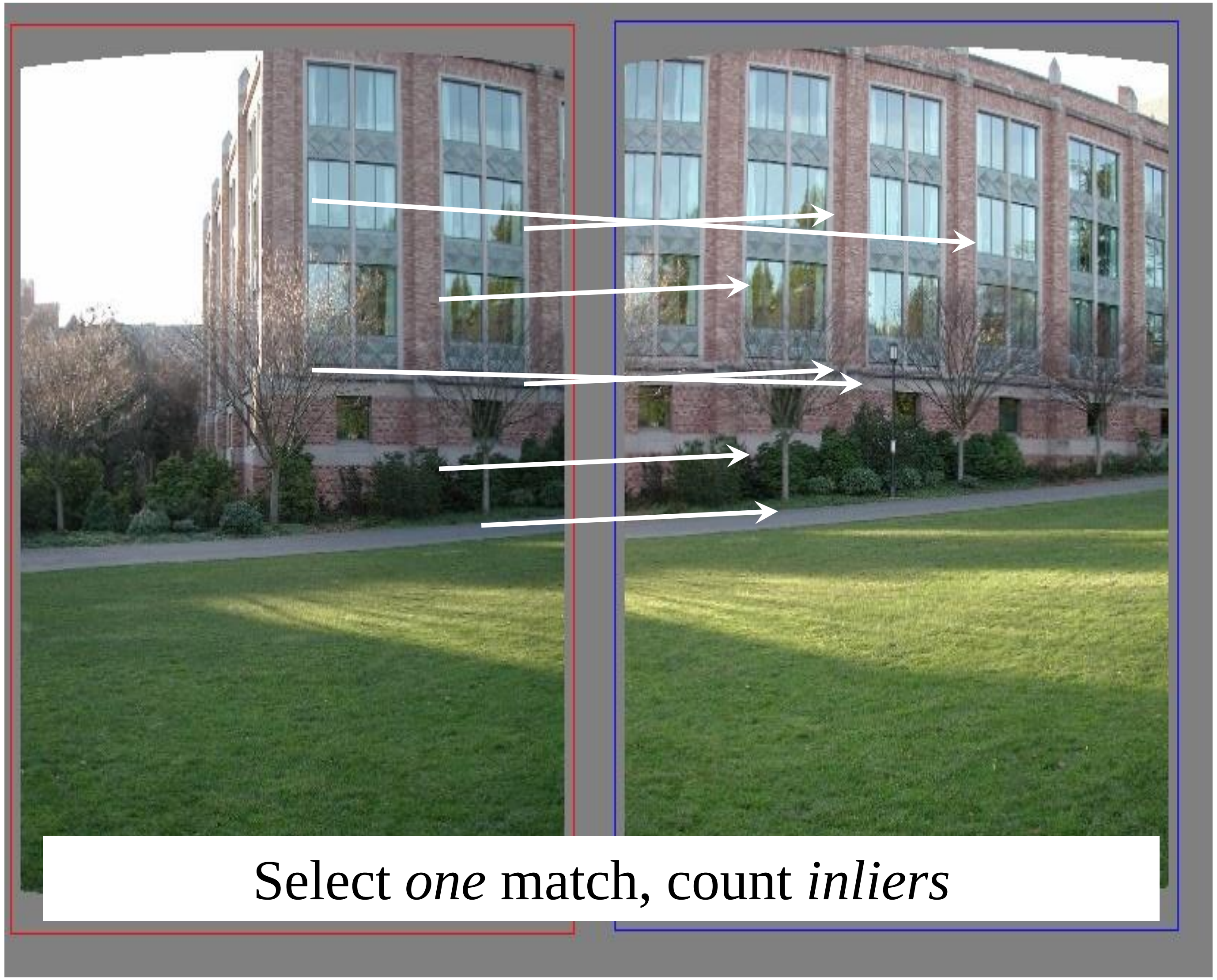
- Even after filtering out ambiguous matches, the set of putative matches still contains a very high percentage of outliers

- Solution: RANSAC
- 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
- At the end, keep the transformation with the largest number of inliers

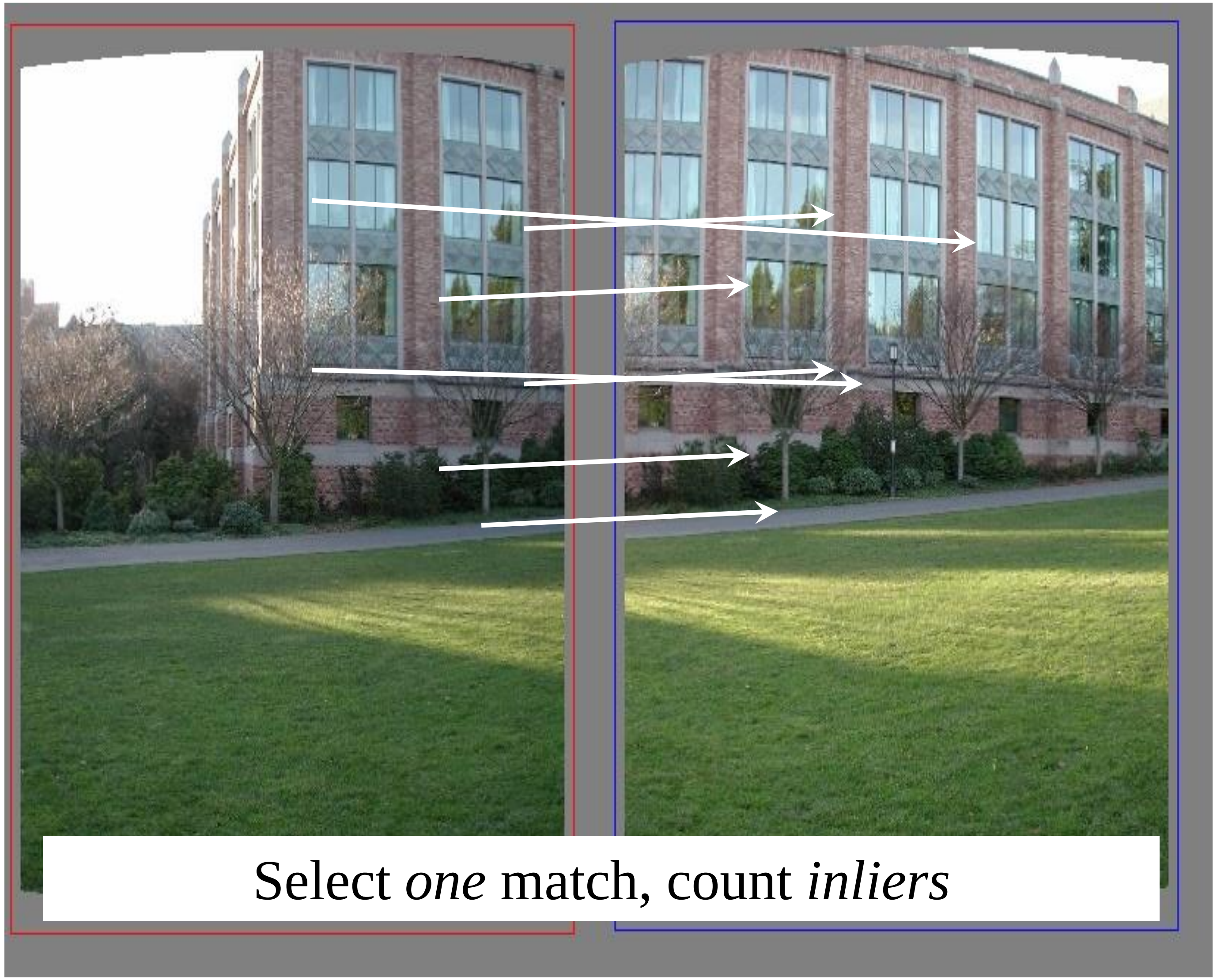
RANSAC example: Translation



RANSAC example: Translation

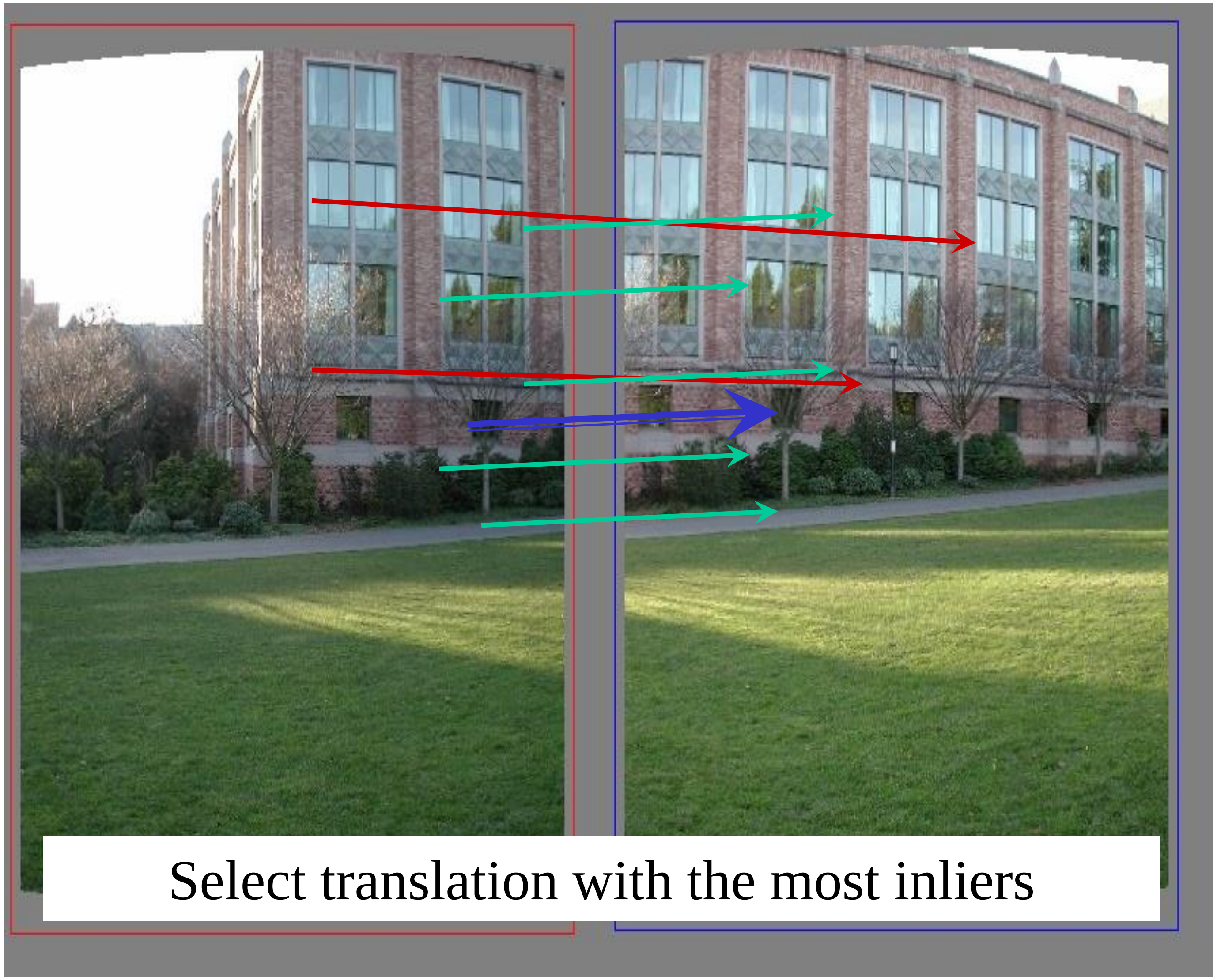


RANSAC example: Translation



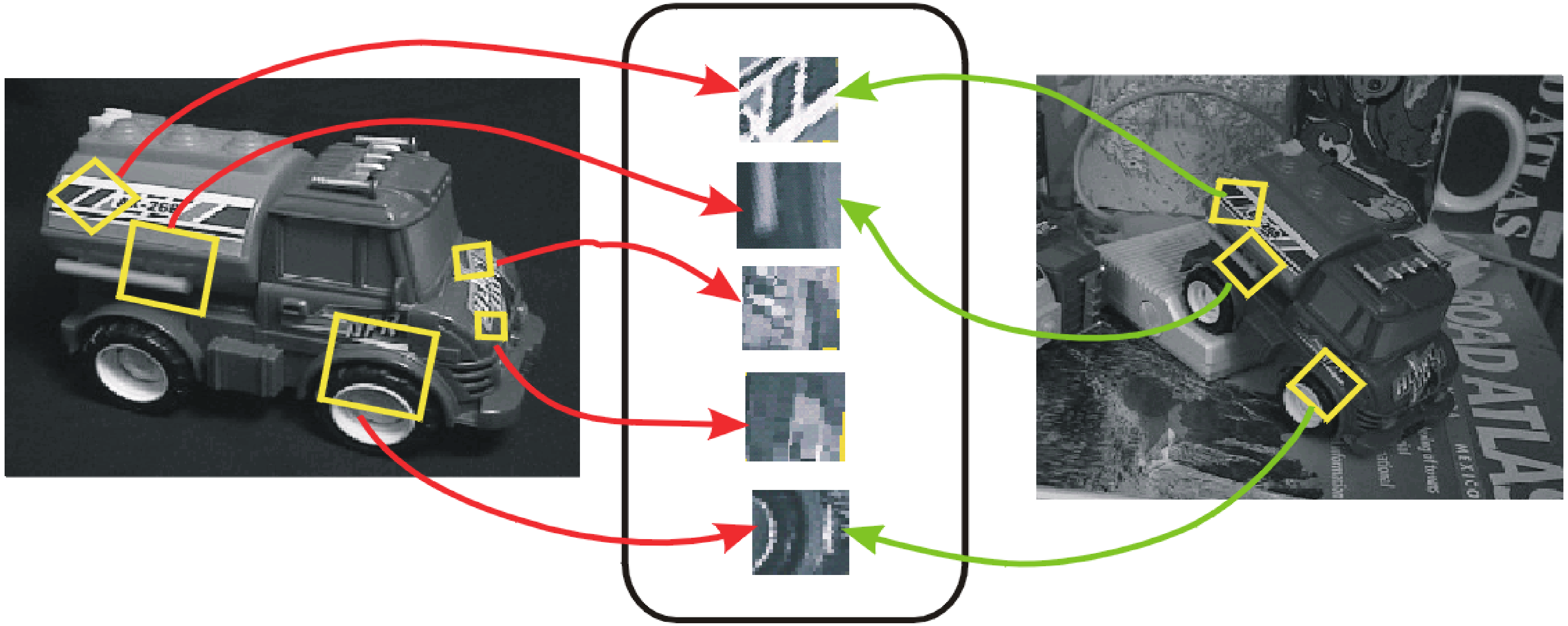
Select *one* match, count *inliers*

RANSAC example: Translation



Alternative for robust alignment: Hough voting

- A single SIFT match can vote for translation, rotation, and scale parameters of a transformation between two images
  - Votes can be accumulated in a 4D Hough space with large bins
  - Clusters of matches falling into the same bin should undergo a more precise verification procedure



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

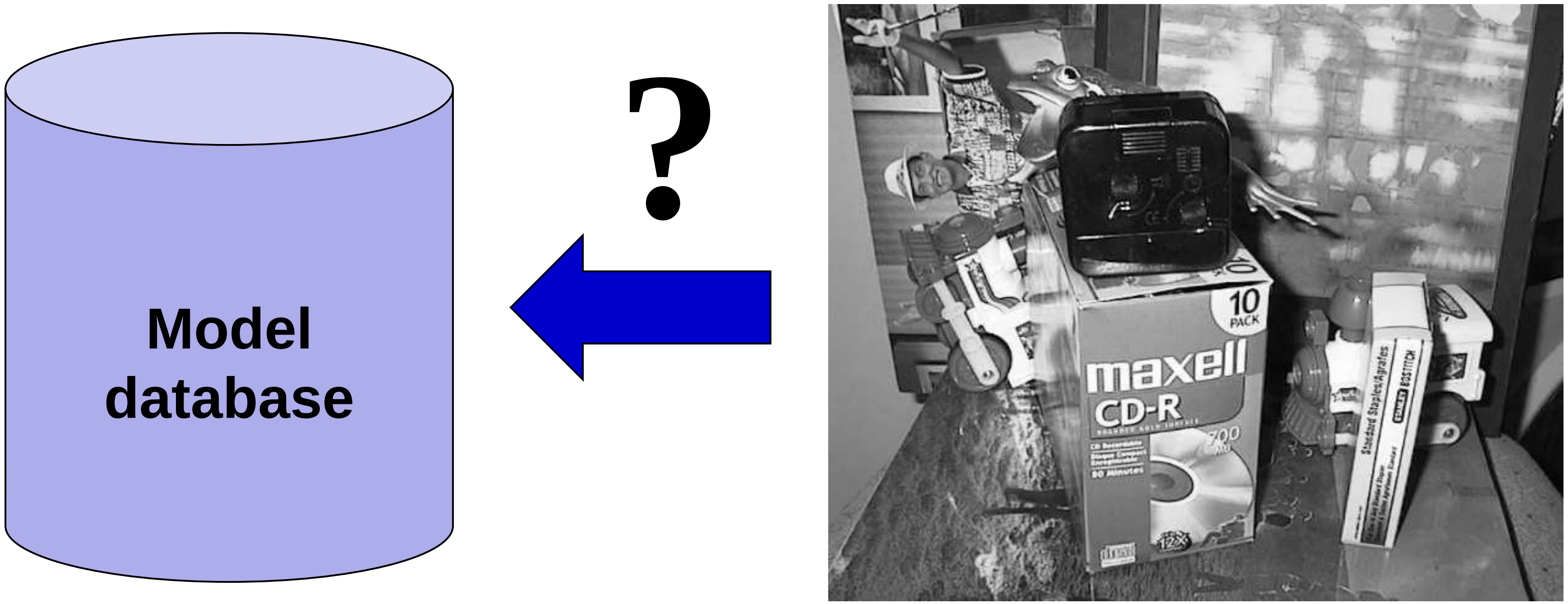
Alignment: Overview

- Motivation
- Fitting of transformations
  - Affine transformations
  - Homographies
- Robust alignment
  - Descriptor-based feature matching
  - RANSAC
- Large-scale alignment
  - Inverted indexing
  - Vocabulary trees

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

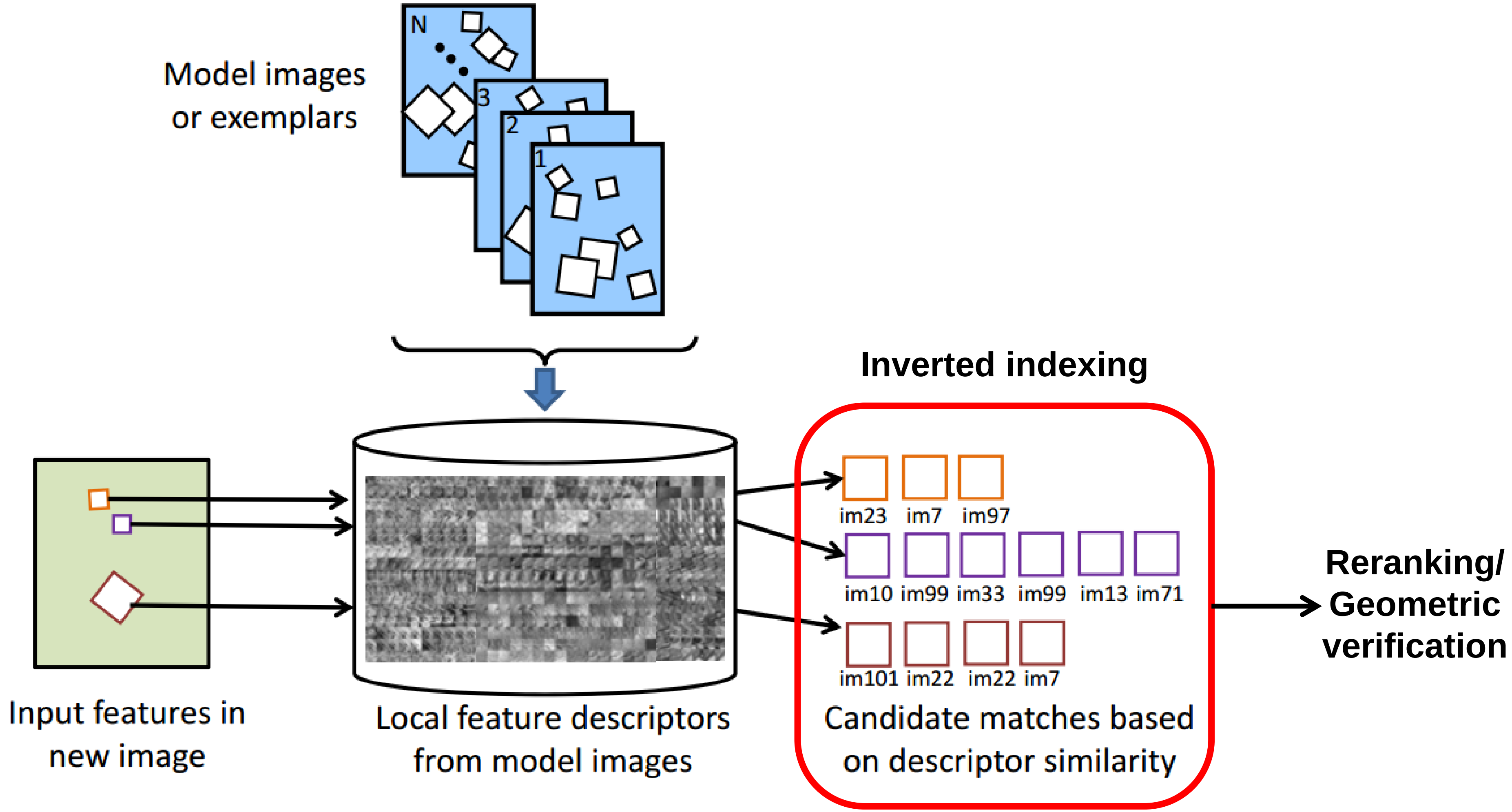
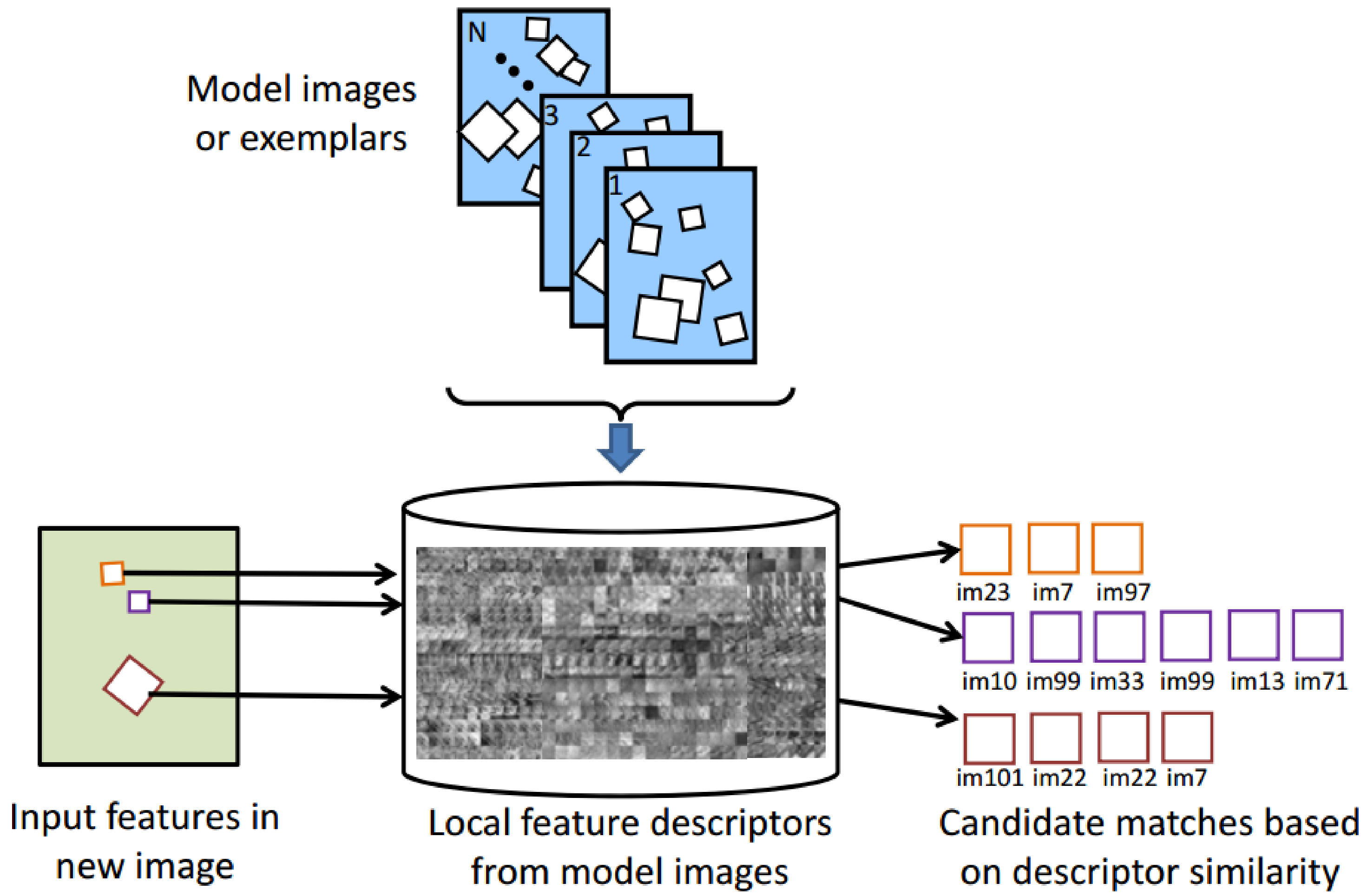


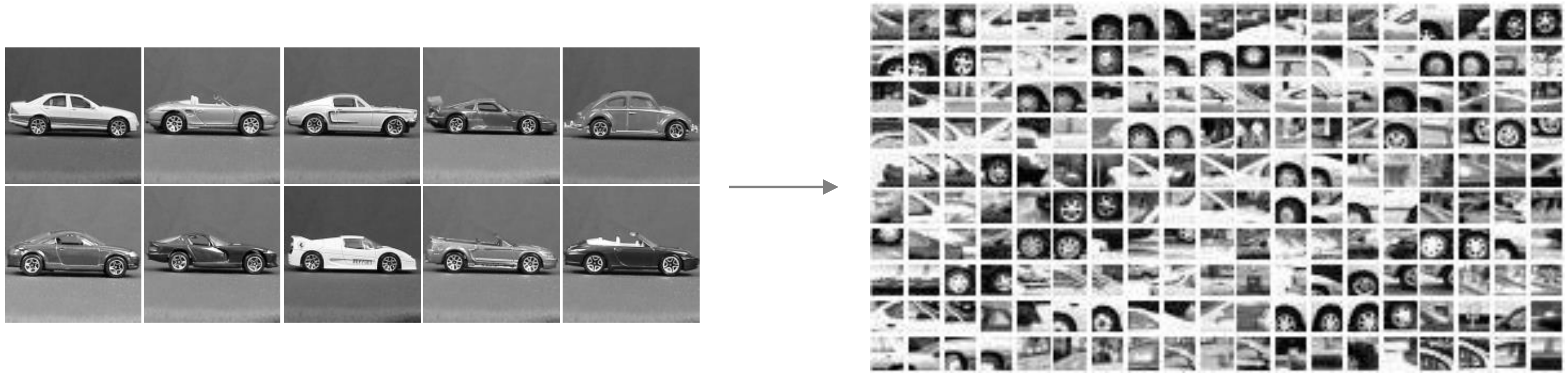
Figure from: Kristen Grauman and Bastian Leibe, Visual Object Recognition, Synthesis Lectures on Artificial Intelligence and Machine Learning, April 2011, Vol. 5, No. 2, Pages 1-181

How to do the indexing?

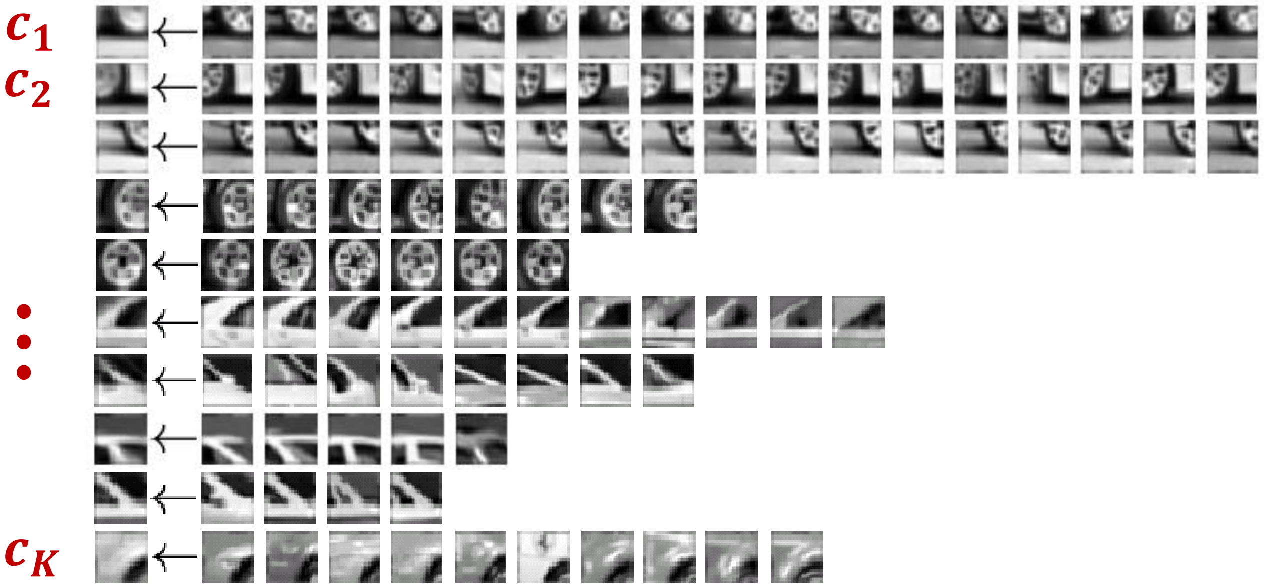


- Idea: find a set of *visual codewords* to which descriptors can be *quantized*

Recall: Visual codebook for implicit shape models



Appearance codebook



B. Leibe, A. Leonardis, and B. Schiele, [Combined Object Categorization and Segmentation with an Implicit Shape Model](), ECCV Workshop on Statistical Learning in Computer Vision 2004

K-means clustering

- We want to find K cluster centers and an assignment of points to cluster centers to minimize the sum of squared Euclidean distances between each point and its assigned cluster center:

$$\sum_i \sum_k a_{ik} \| \boldsymbol{x}_i - \boldsymbol{c}_k \|^2$$

Sum over all points

Sum over all clusters

Point $\boldsymbol{x}_i$ is assigned to cluster k

Center of cluster k

K-means clustering

- We want to find K cluster centers and an assignment of points to cluster centers to minimize the sum of squared Euclidean distances between each point and its assigned cluster center:

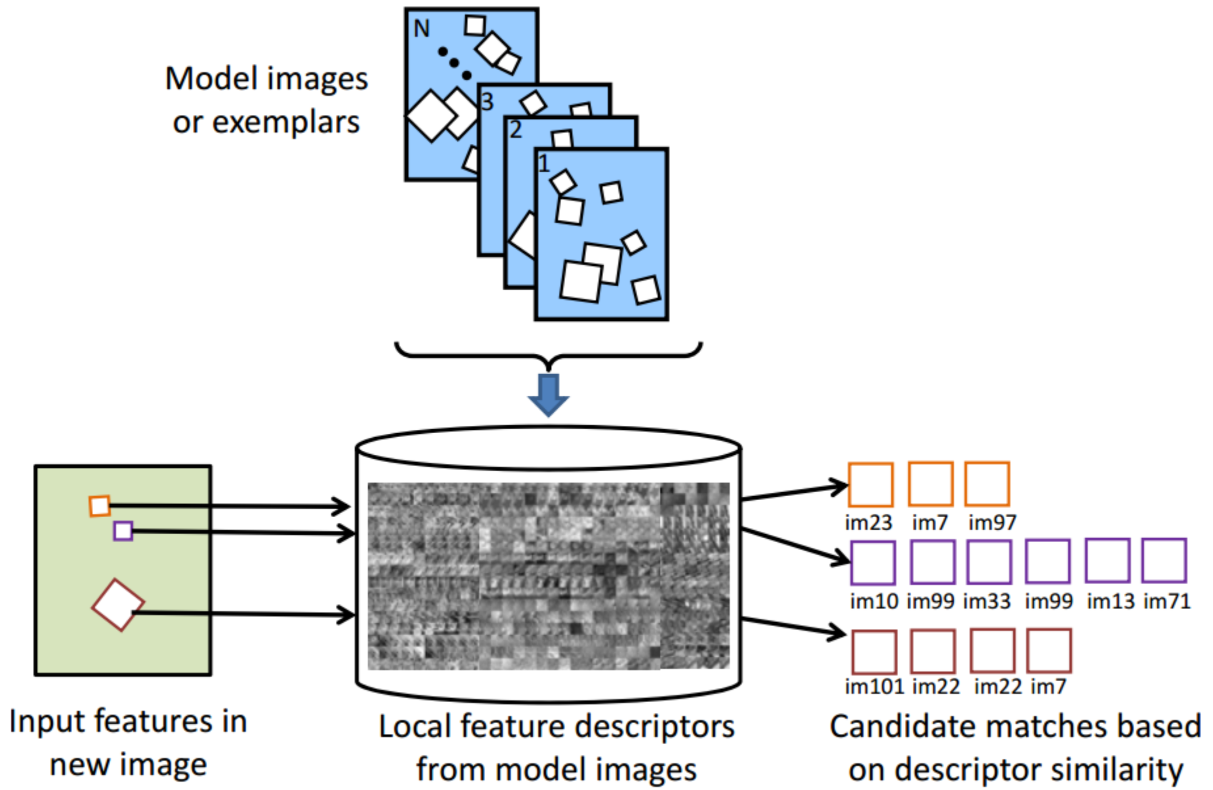
$$\sum_i \sum_k a_{ik} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2$$

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

K-means example

Source

How to do the indexing?



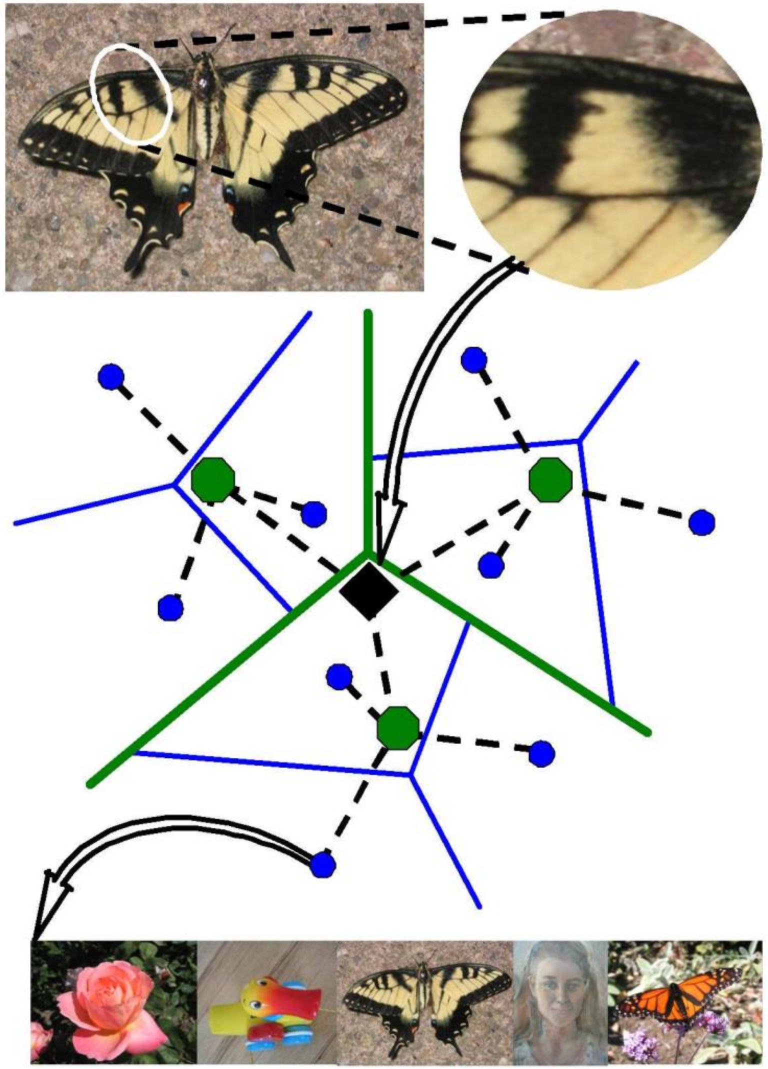
- Cluster descriptors in the database to form codebook
- At query time, quantize descriptors in query image to nearest codevectors
- Problem solved?

Efficient indexing technique: Vocabulary trees

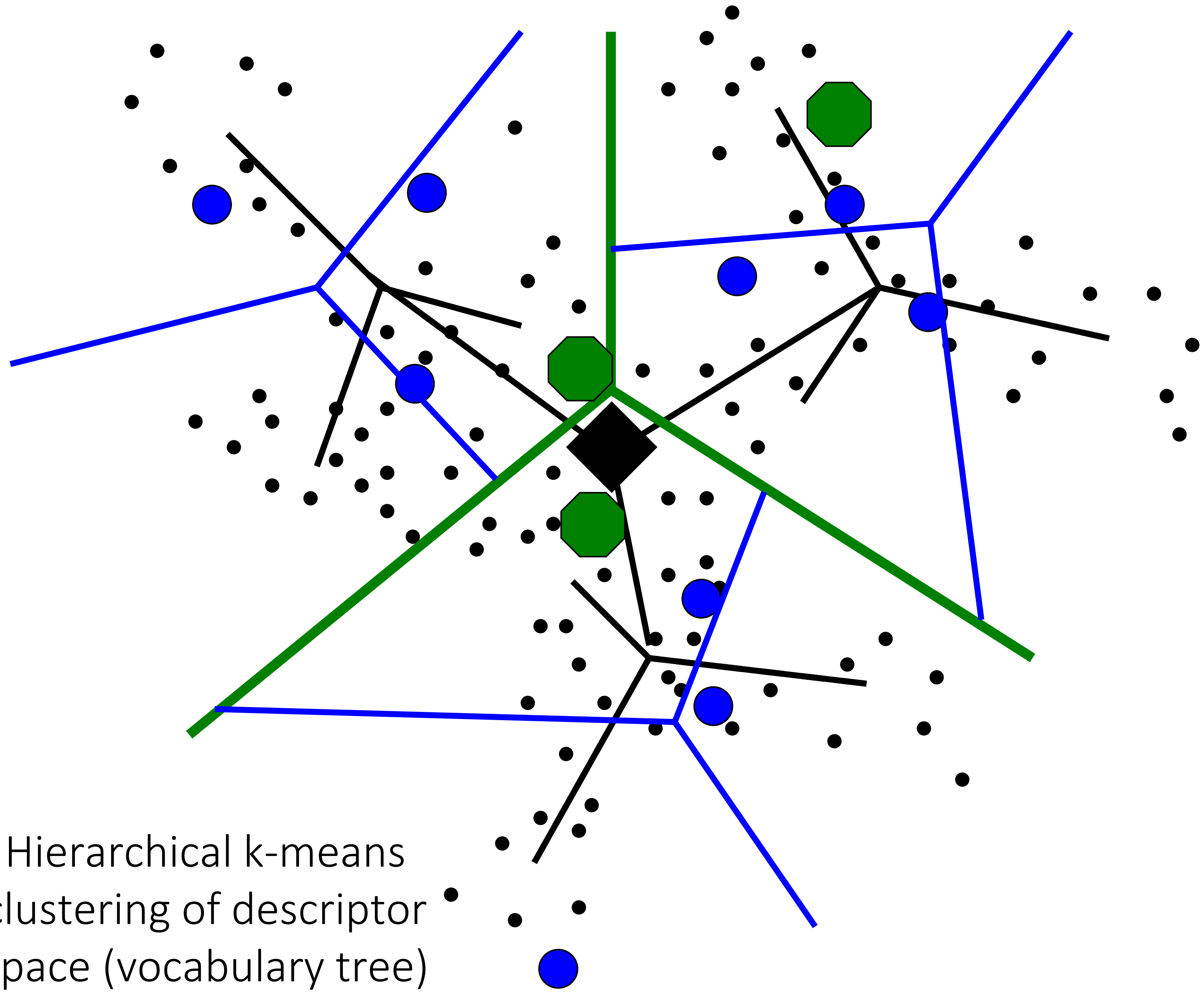
Test image

Vocabulary tree
with inverted
index

Database

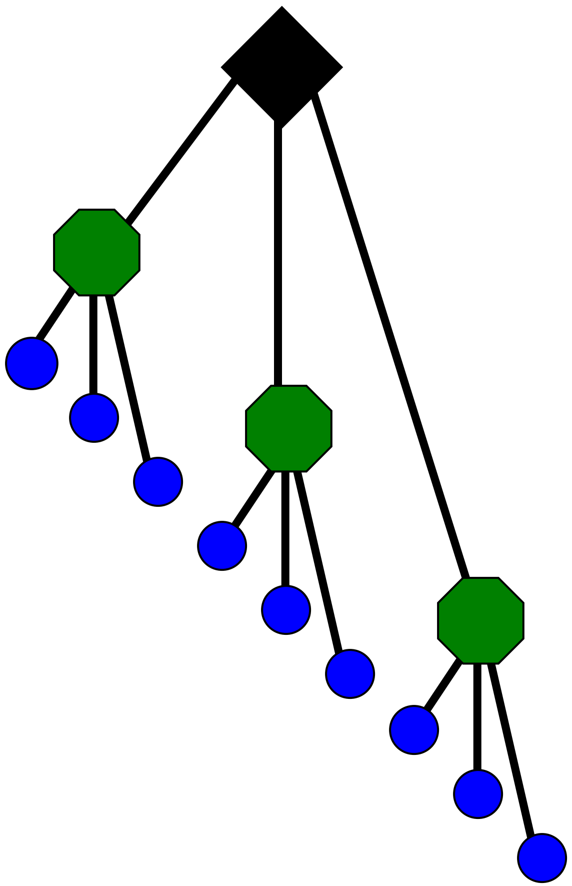


D. Nistér and H. Stewénius, [Scalable Recognition with a Vocabulary Tree](), CVPR 2006



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

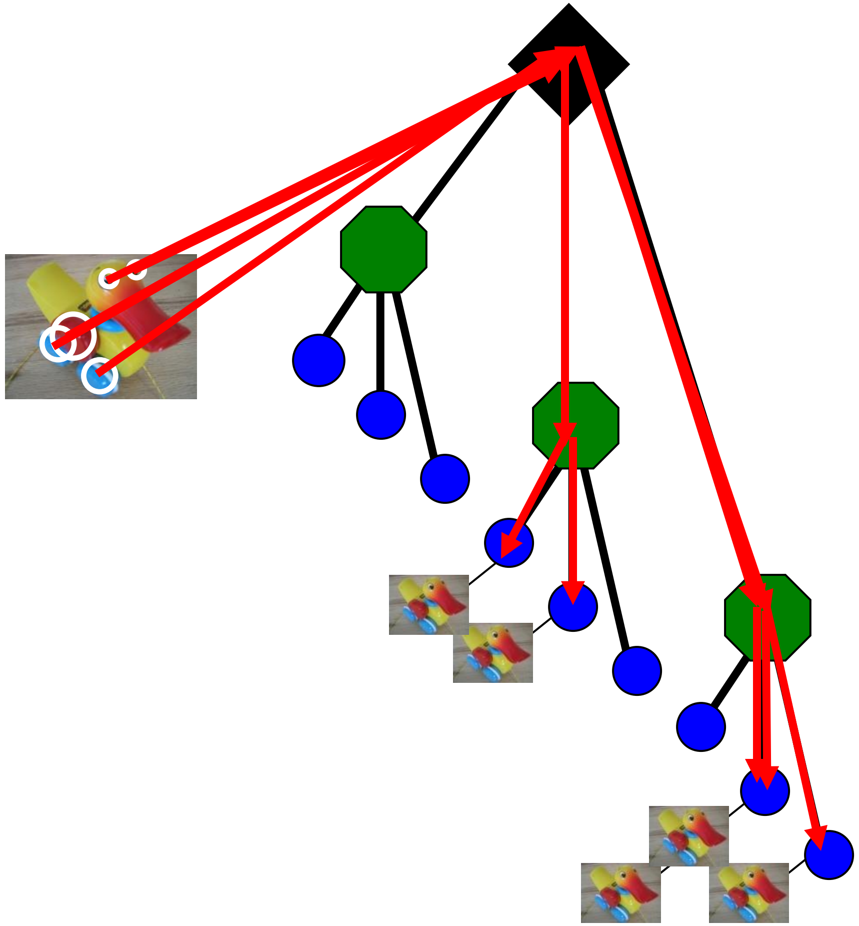
Slide credit: D. Nister



Vocabulary tree/inverted index

Slide credit: D. Nister

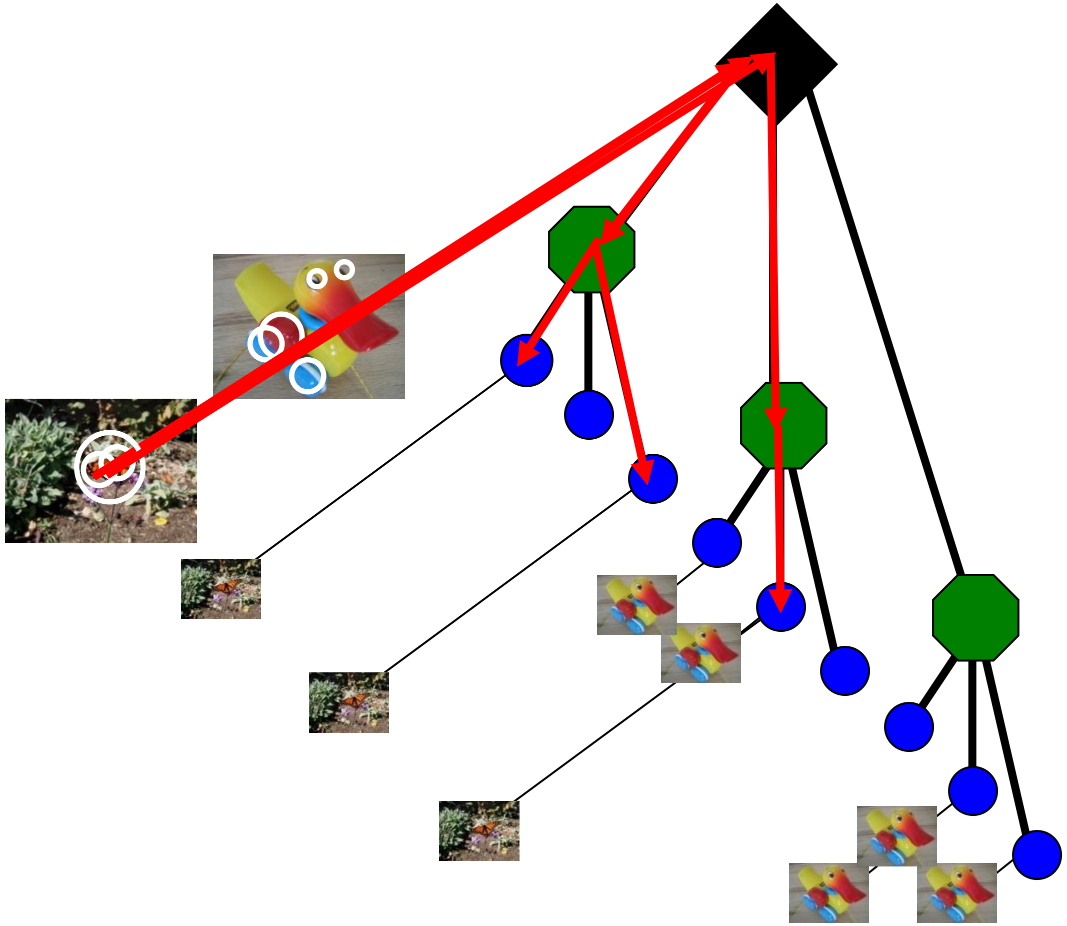
Model images



Populating the vocabulary tree/inverted index

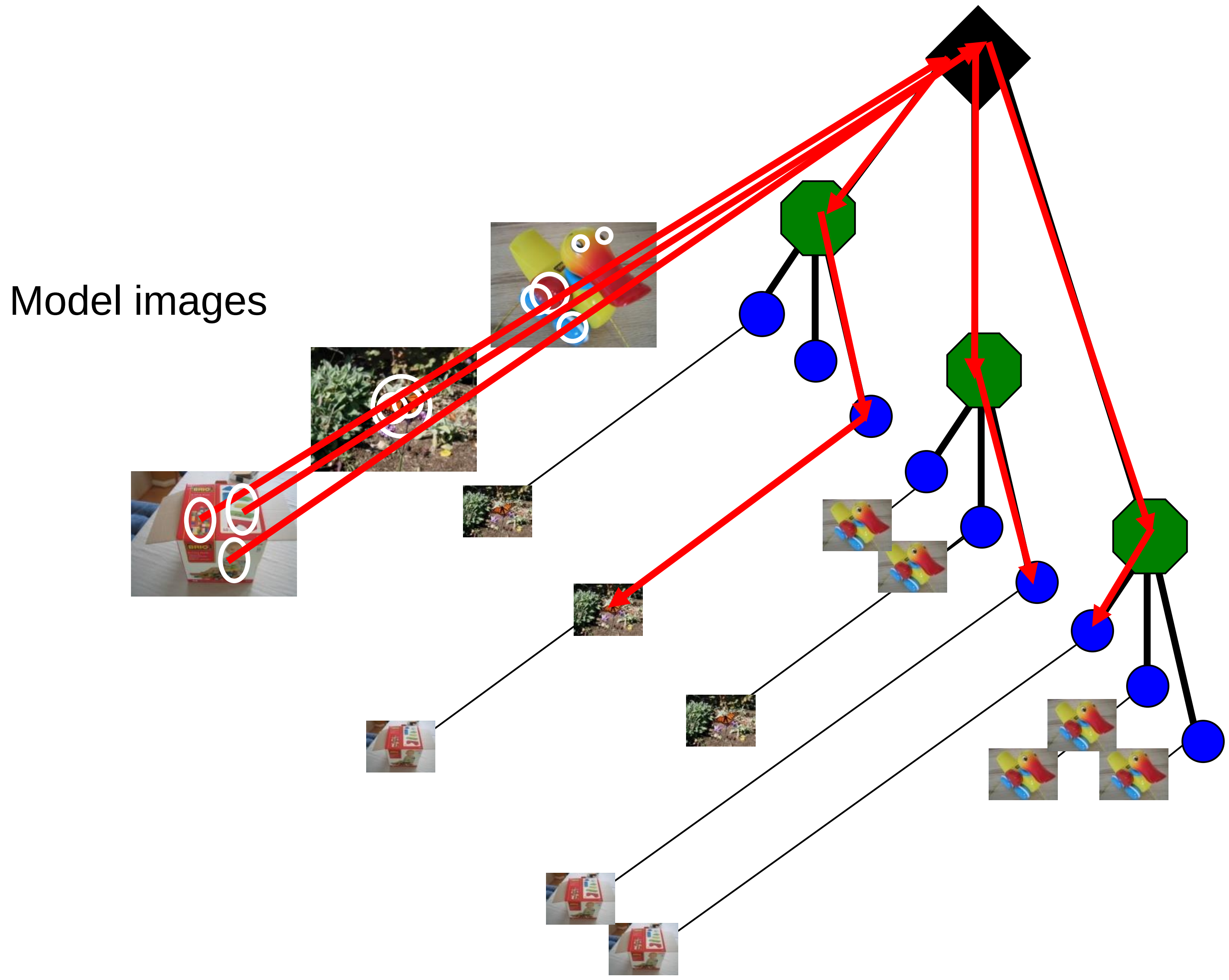
Slide credit: D. Nister

Model images



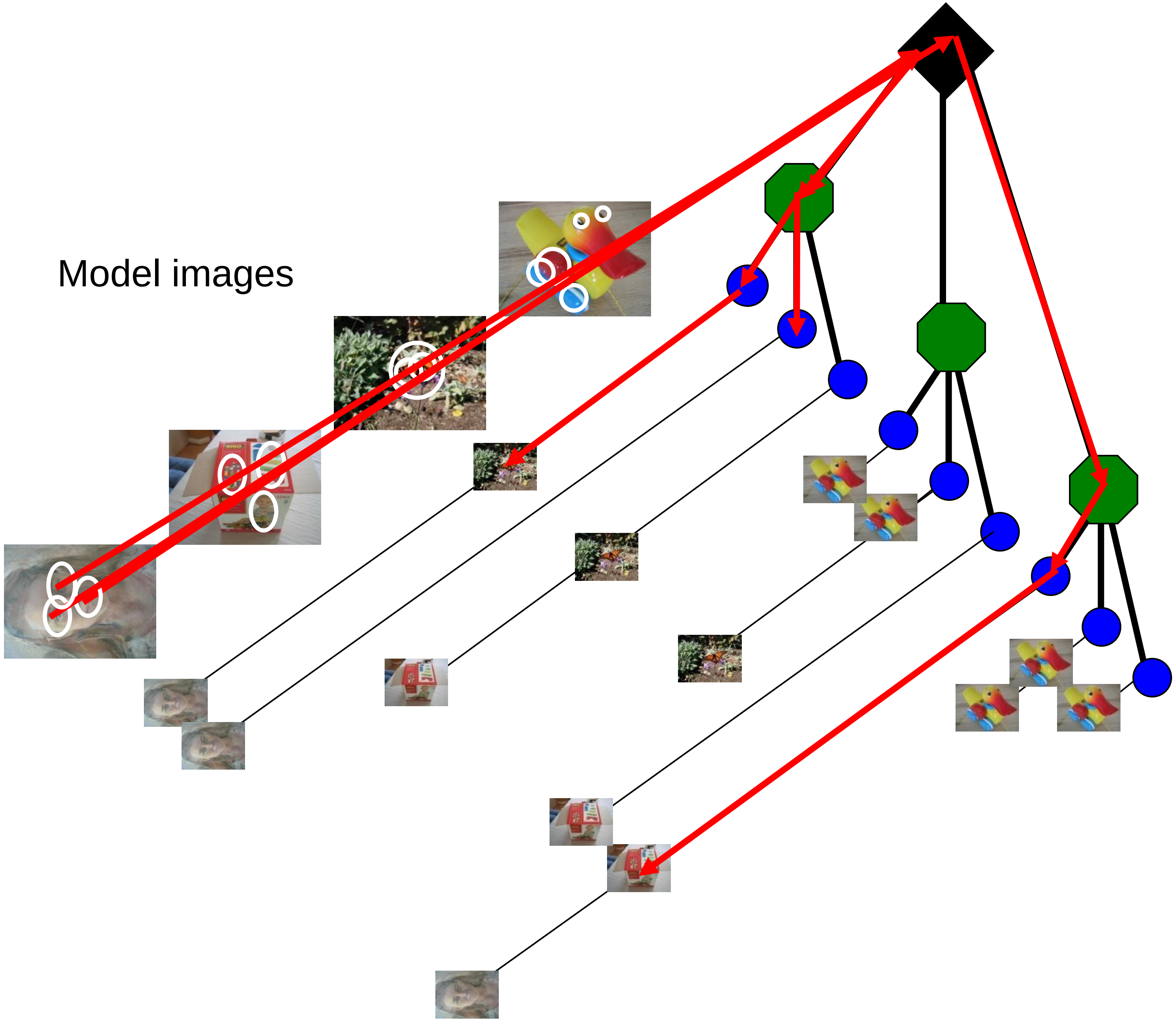
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



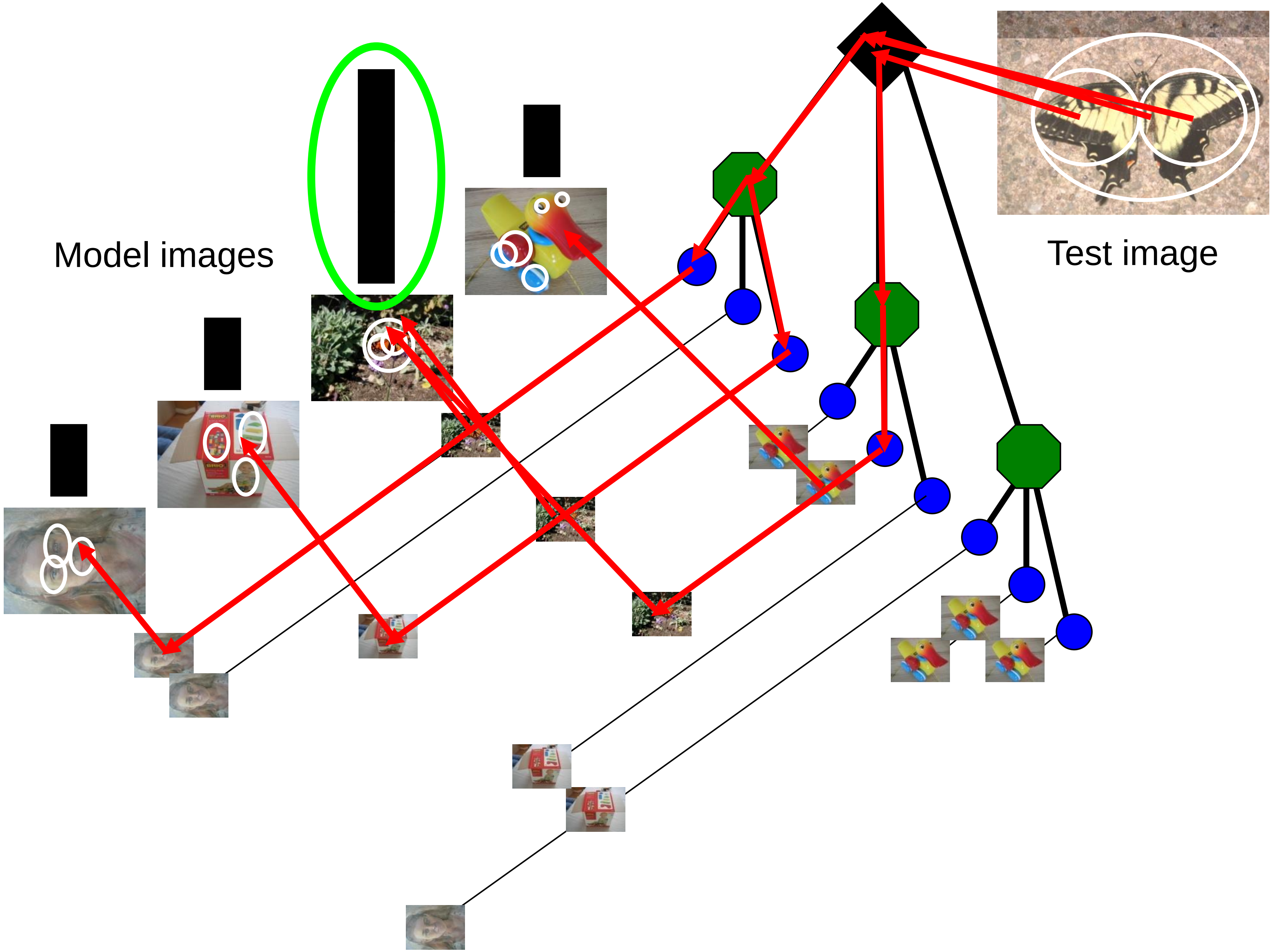
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



Populating the vocabulary tree/inverted index

Slide credit: D. Nister



Looking up a test image

Slide credit: D. Nister