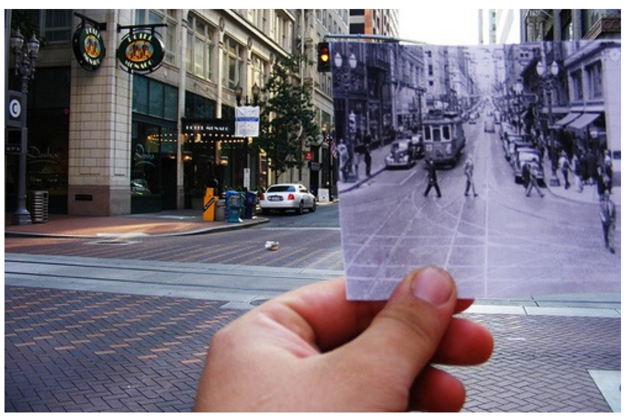
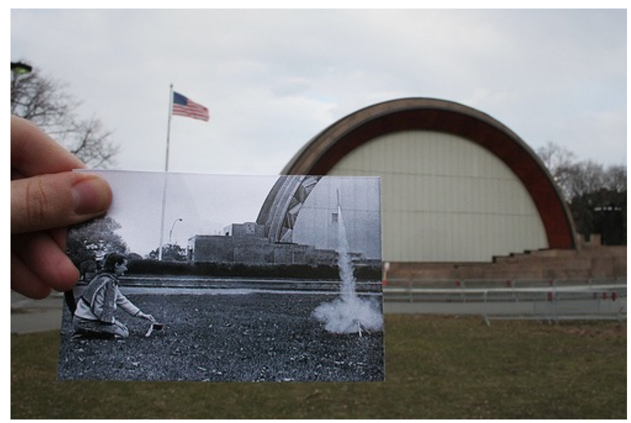
Image alignment

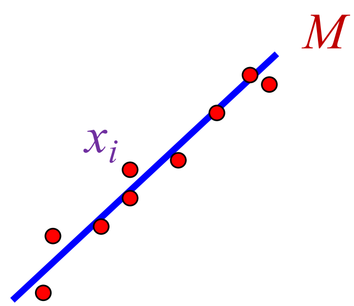


Source

Alignment: Fitting of transformations

- Previously: fitting a model to features in one image

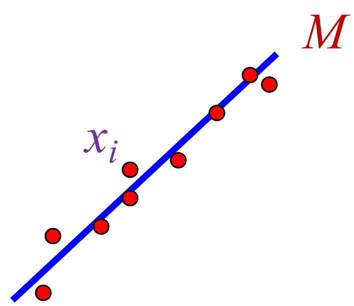


- Given: points $x_1, \dots, x_n$
- Find: model M that minimizes

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

Alignment: Fitting of transformations

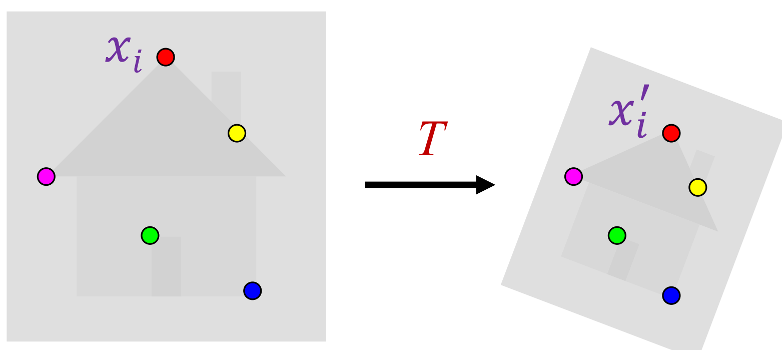
- Previously: fitting a model to features in one image



- 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: Fitting of transformations

- General problem:
  - Compute the best transformation of a given class from one set of points to another
- Cases
  - 2D points in images
  - 3D points
  - Different kinds of transformation
    - Some allow closed form solutions
  - With correspondences; without correspondences
  - Robust transformations

Other registration applications

3D point clouds (depth sensors measure 3D position of points)

- Given a 3D model of an object, determine transformation from object to points
- Given two views (back and side, say) of object, align them

Localization

- Given a map of an environment, determine transformation from observations to map
  - This tells you where you are
  - Observations are 3D points, but could be more interesting

Mapping

- Given a bunch of observations from different locations, turn them into a map
  - By registering them to one another

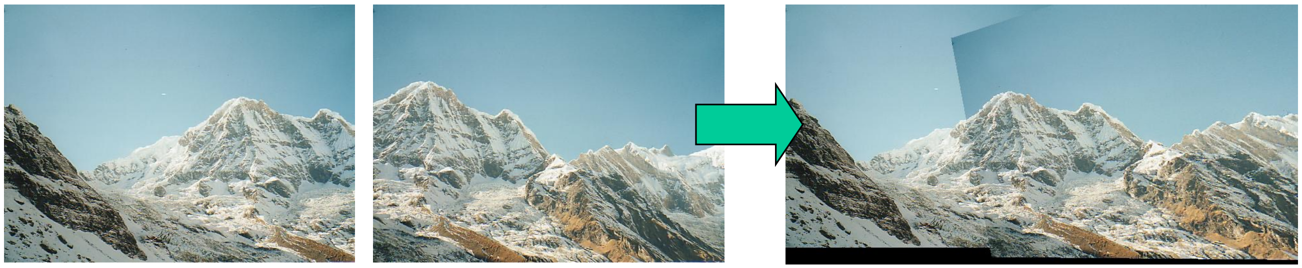
Calibrating a camera

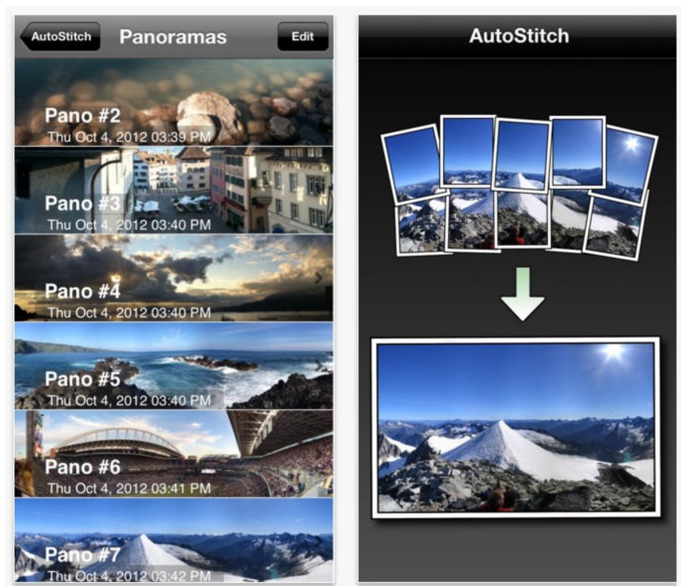
- Camera sees a set of points on a known object; figure out a bunch of camera parameters (later)

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

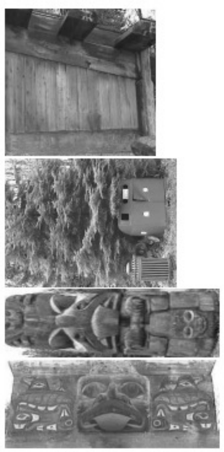




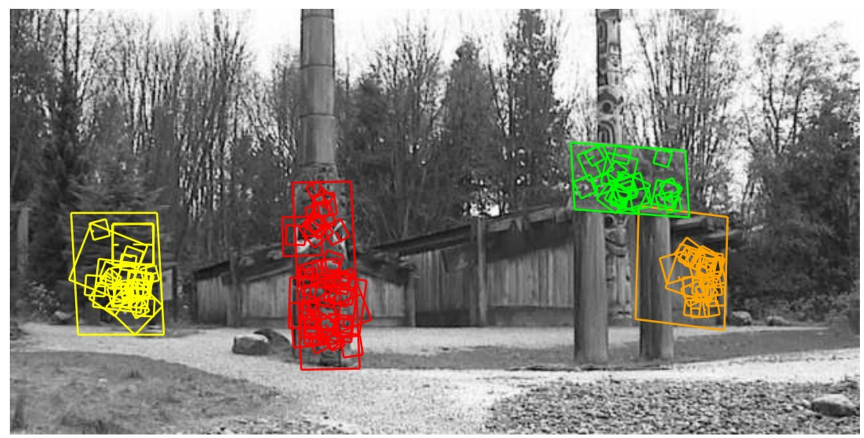
http://matthewalunbrown.com/autostitch/autostitch.html

Alignment applications: Instance recognition

Model images



Test image



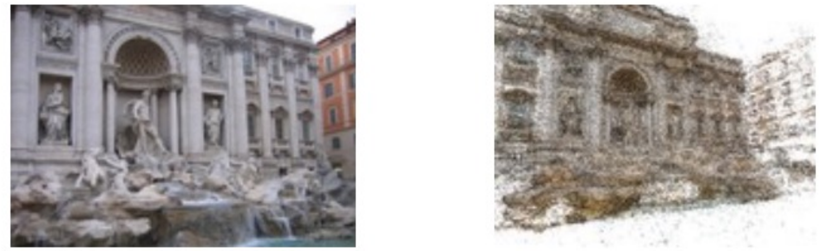
David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV* 60 (2), pp. 91-110, 2004

Alignment applications: Instance recognition

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

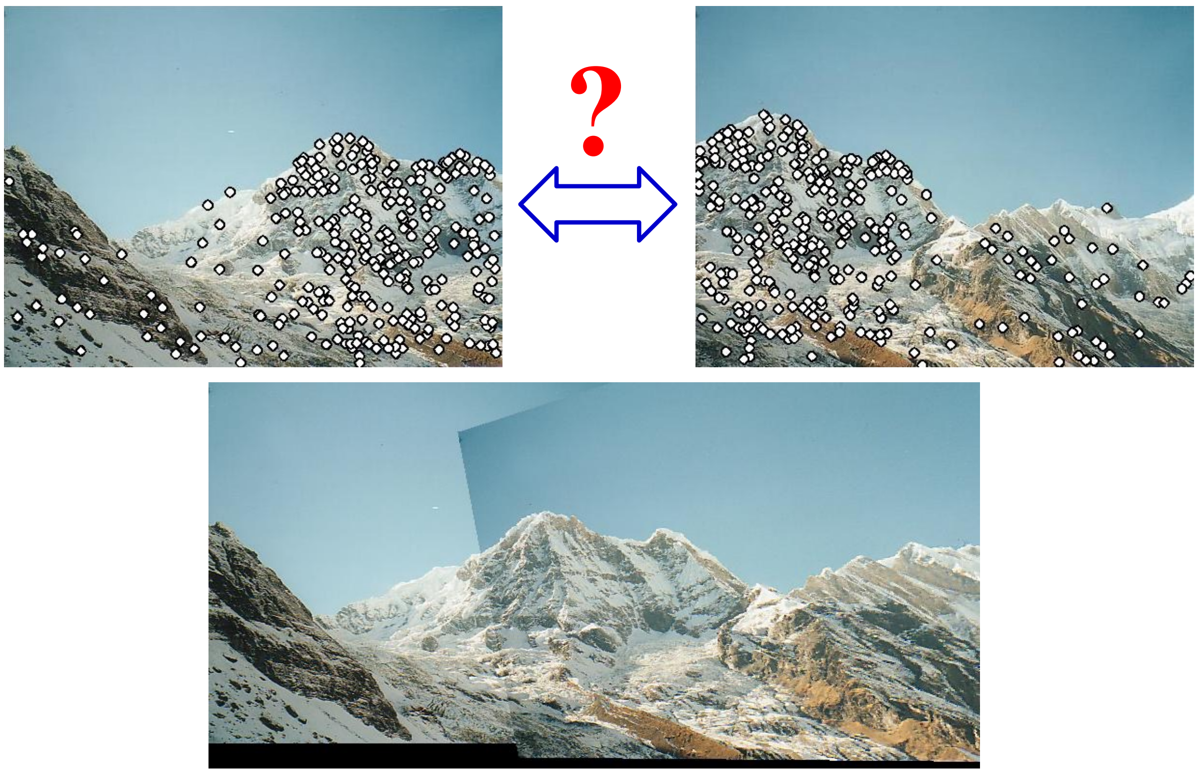
Pantheon: 1,032 images, 530,076 points

Hall of Maps: 275 images, 230,182 points

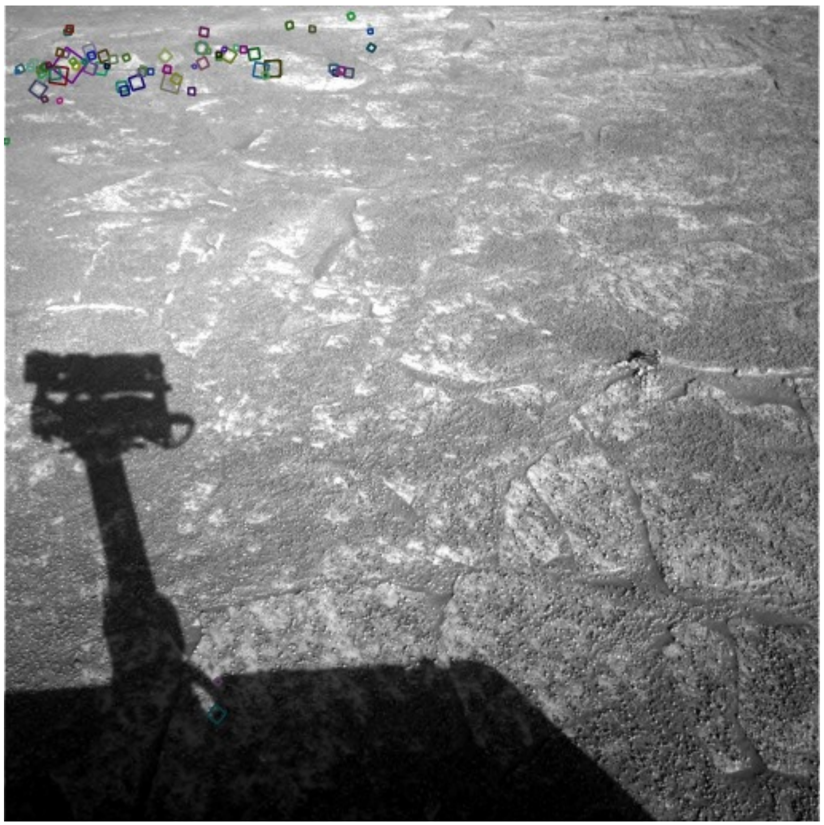
S. Agarwal et al. Building Rome in a Day. ICCV 2009

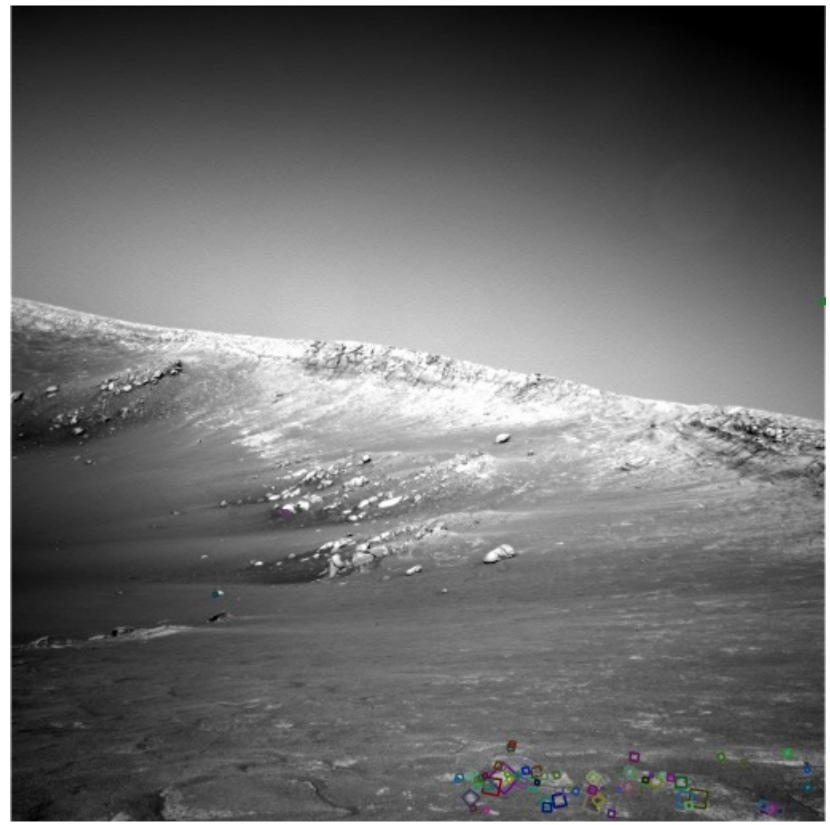
Feature-based alignment

- 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

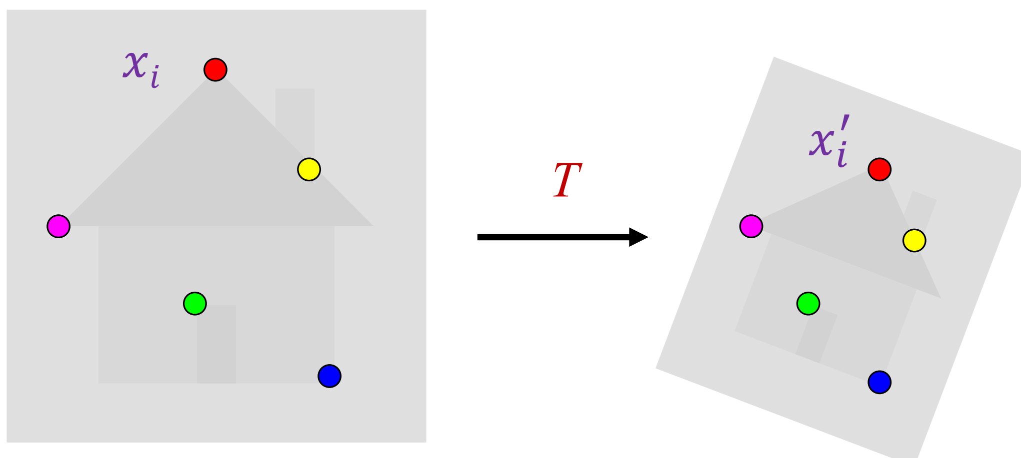
Alignment: Overview

- Motivation
- Fitting of transformations
  - Affine transformations
  - Homographies

Alignment: Fitting of transformations

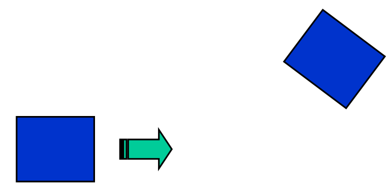
- Given: matches $(x_1, x_1'), \ldots, (x_n, x_n')$
- Find: transformation T that minimizes

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

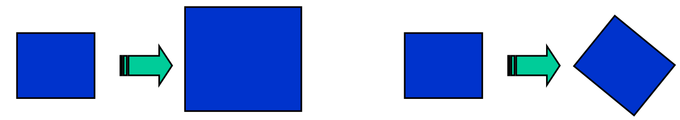


2D transformation models

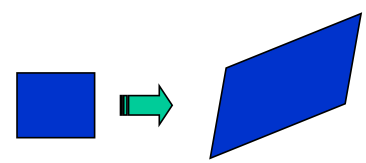
- Euclidean (rotation, translation)



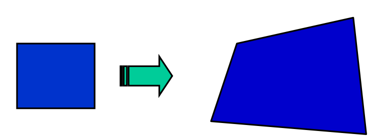
- Similarity (translation, scale, rotation)



- Affine



- Projective (homography)



Transformations

Affine transformations transform $\mathbf{y}$ to $\mathbf{x} = \mathcal{M}\mathbf{y} + \mathbf{t}$, where $\mathcal{M}$ has non-zero determinant. Some kinds of affine transform have specialized names:

- **Translation:** when $\mathcal{M}$ is the identity.
- **Rotation:** when $\mathbf{t} = \mathbf{0}$; $\mathcal{M}^T\mathcal{M}$ is the identity and $\mathcal{M}$ has positive determinant.
- **Homogenous scaling:** when $\mathcal{M}$ is σ times the identity, and $\sigma \neq 0$.
- **Scaling:** when $\mathcal{M}$ is diagonal.
- **Euclidean or rigid body:** when $\mathcal{M}^T\mathcal{M}$ is the identity and $\mathcal{M}$ has positive determinant.

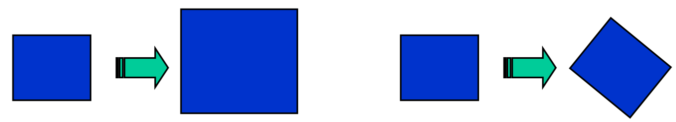
2D transformation models

Euclidean (rotation and translation)

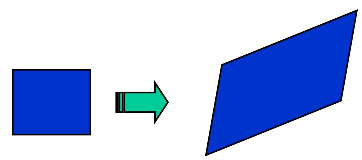
A closed form solution to least squares problem is known. It’s in the notes. Won’t do that here, as it’s a bit elaborate.

2D transformation models

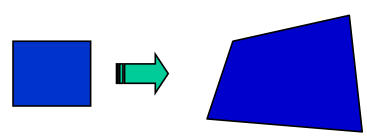
- Similarity (translation, scale, rotation)



- Affine

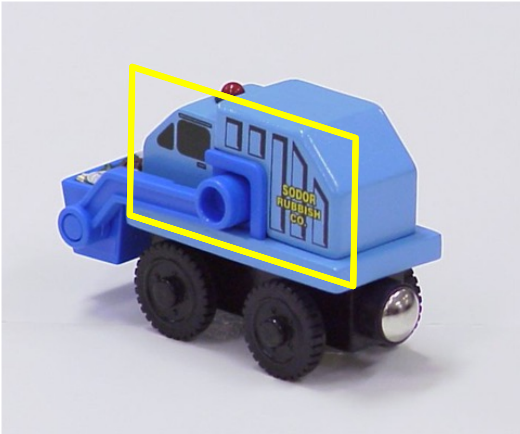


- Projective (homography)



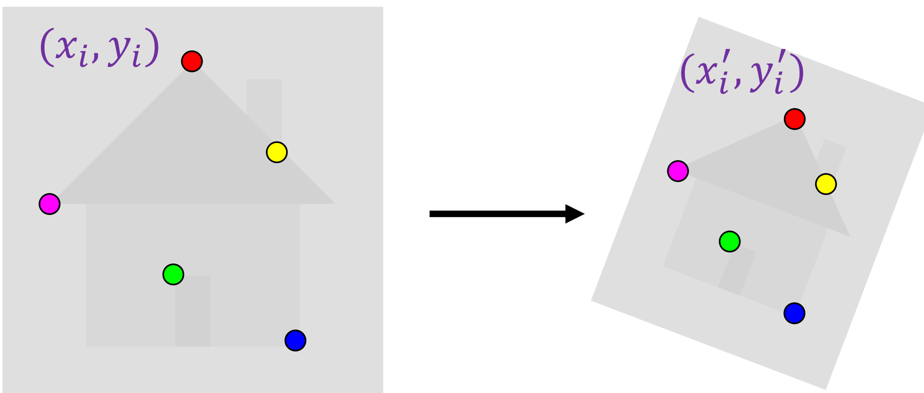
Let’s start with affine transformations

- 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?



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

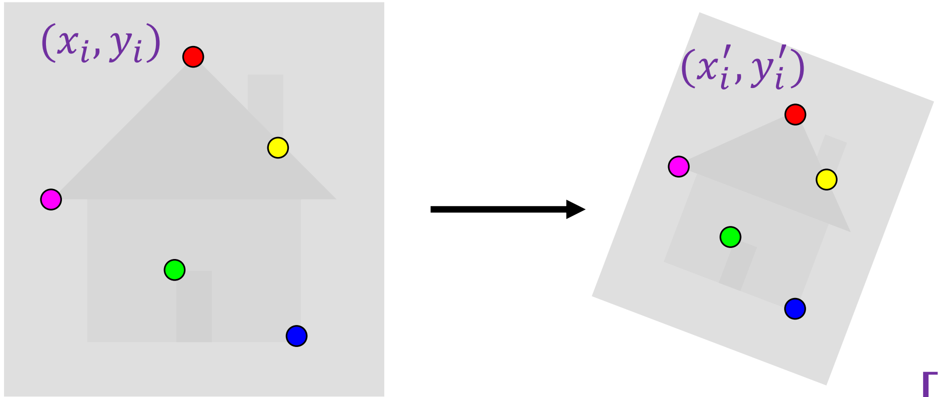
$\boldsymbol{x}_i'$ $\quad\quad$ $\boldsymbol{M}$ $\quad\quad$ $\boldsymbol{x}_i$ $\quad\quad$ $\boldsymbol{t}$

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?

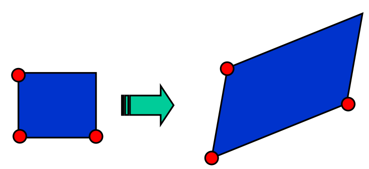


$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

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

Fitting an affine transformation

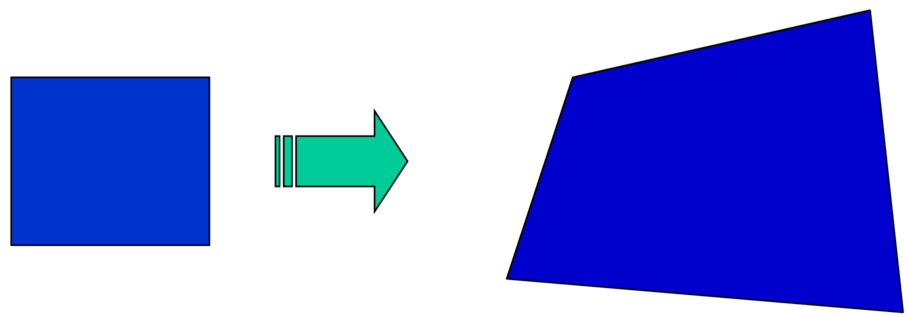
- How many matches do we need to solve for the transformation parameters?



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

Fitting a homography

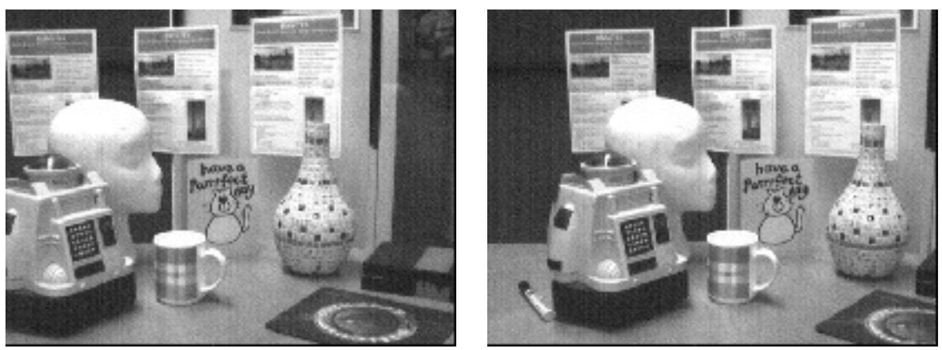
- A *homography* is a plane projective transformation (transformation taking a quad to another arbitrary quad)



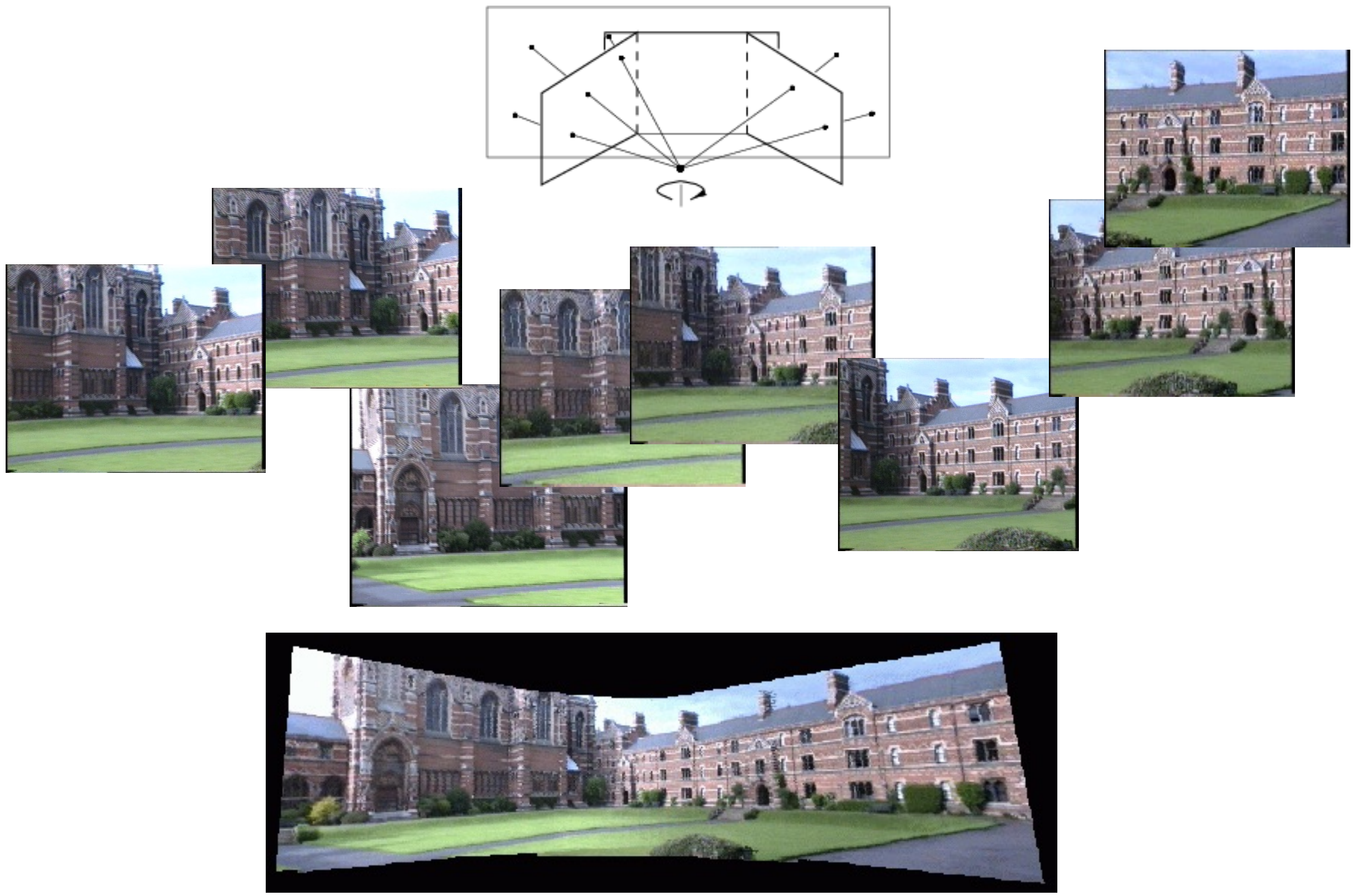
Homography in the real world

- 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

Projective transformations in general

Projective transformations are a new class. We will see much more of the geometry underlying projective transformations later. A projective transformation in N dimensions is given by an $N+1 \times N+1$ dimensional matrix $\mathcal{P}$ with non-zero determinant. There are a number of different ways of representing the effect of a projective transformation. For now, we will write an N D projective transformation as

$$\begin{pmatrix} \mathcal{M} & \mathbf{v} \\ \mathbf{m}_{N+1}^T & v_{N+1} \end{pmatrix} = \begin{pmatrix} \mathbf{m}_1^T & v_1 \\ \ldots & \\ \mathbf{m}_N^T & v_N \\ \mathbf{m}_{N+1}^T & v_{N+1} \end{pmatrix}$$

Homography is the case where N=2

where $\mathcal{M}$ is $N \times N$, and the vectors are $N \times 1$. This transformation takes $\mathbf{y}$ to

$$\mathbf{x} = \begin{bmatrix} \frac{\mathbf{m}_1^T \mathbf{y} + m_1}{\mathbf{m}_{N+1}^T \mathbf{y} + m_{N+1}} \\ \frac{\mathbf{m}_2^T \mathbf{y} + m_2}{\mathbf{m}_{N+1}^T \mathbf{y} + m_{N+1}} \\ \ldots \\ \frac{\mathbf{m}_N^T \mathbf{y} + m_N}{\mathbf{m}_{N+1}^T \mathbf{y} + m_{N+1}} \end{bmatrix}.$$

Notice that there could be a divide by zero issue here. For the moment, we will ignore this and assume it never happens. In fact, a great deal of interesting geometry follows from paying attention to this issue, as we shall see in Chapter 34.2.

Fitting a homography

We now work with points on the plane, and allow the transformation to be a homography. Solving for a homography requires solving an optimization problem, but estimating a homography from data is useful, and relatively easy to do. We can't recover the translation component from centers of gravity (exercises **TODO: homography exercise**). Write m_{ij} for the i, j'th element of matrix $\mathcal{M}$. In affine coordinates, a homography $\mathcal{M}$ will map $\mathbf{y}_i = (y_{i,x}, y_{i,y})$ to $\mathbf{x}_i = (x_{i,x}, x_{i,y})$ where

$$x_{i,x} = \frac{m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13}}{m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}} \text{ and } x_{i,y} = \frac{m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23}}{m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}} \quad (12.14)$$

Fitting a homography

Write $\mathcal{M}(\mathbf{y})$ for the result of applying the homography to $\mathbf{y}$ as above. In most cases of interest, the coordinates of the points are not measured precisely, so we observe $\mathbf{x}_i = \mathcal{M}(\mathbf{y}_i) + \xi_i$, where ξ_i is some noise vector drawn from an isotropic normal distribution with mean $\mathbf{0}$ and covariance Σ. Again, assume that the noise is isotropic, and so that $\Sigma = \sigma^2\mathcal{I}$. The homography can be estimated by minimizing the negative log-likelihood of the noise, so we must minimize

$$\sum_i w_i \xi_i^T \xi_i \tag{12.15}$$

where

$$\xi_i = \left[\begin{array}{c} x_{i,x} - \frac{m_{11}y_{i,x}+m_{12}y_{i,y}+m_{13}}{m_{31}y_{i,x}+m_{32}y_{i,y}+m_{33}} \\ x_{i,y} - \frac{m_{21}y_{i,x}+m_{22}y_{i,y}+m_{23}}{m_{31}y_{i,x}+m_{32}y_{i,y}+m_{33}} \end{array} \right] \tag{12.16}$$

using standard methods (Levenberg-Marquardt is favored; Chapter 34.2). This approach is sometimes known as *maximum likelihood* . Experience teaches that this optimization is not well behaved without a strong start point.

Fitting a homography: finding a start point

Remember

$$x_{i,x} = \frac{m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13}}{m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}} \text{ and } x_{i,y} = \frac{m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23}}{m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}}$$

So that

Known **Known**

$$x_{i,x}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13} = 0$$

and

Known **Known**

$$x_{i,y}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23} = 0$$

Fitting a homography: finding a start point

$$x_{i,x}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13} = 0$$

and

$$x_{i,y}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23} = 0$$

Each corresponding pair of points yields two homogenous equations

Total of 9 unknowns – can solve up to scale with 4 pairs

But scaling a projective transformation doesn't change the tx!

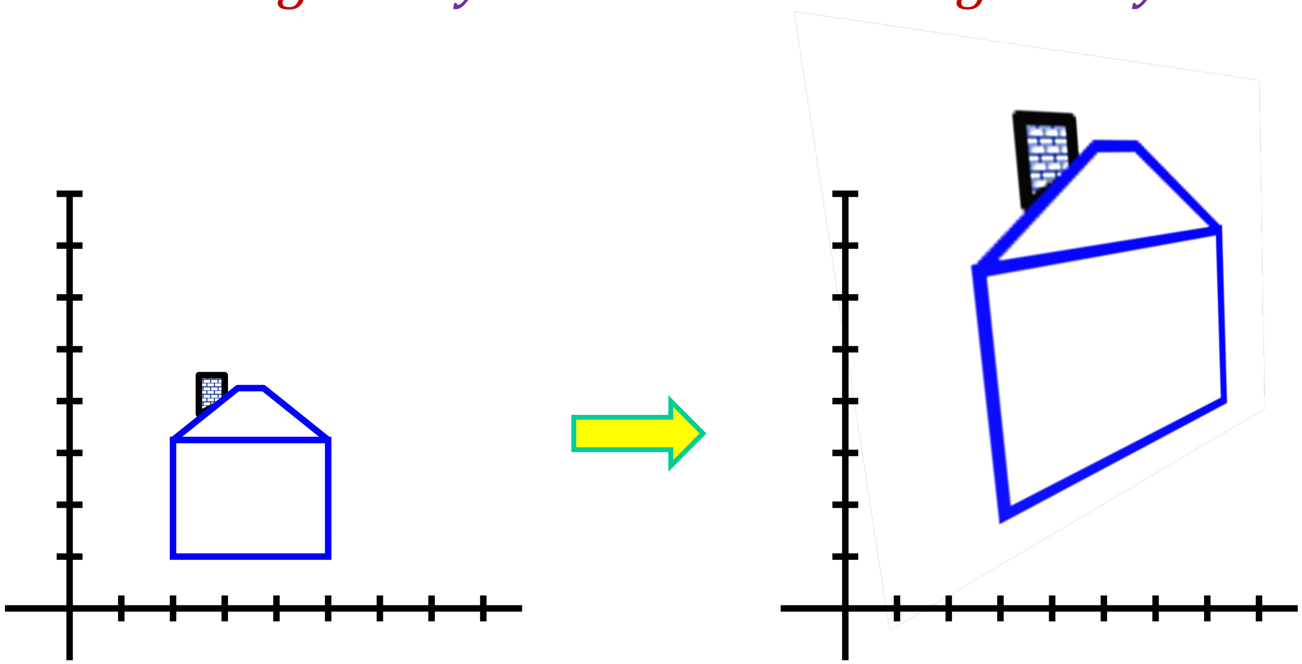
Scaling doesn't change projective transformation

$$x_{i,x} = \frac{m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13}}{m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}} \text{ and } x_{i,y} = \frac{m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23}}{m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}}$$

Fitting a homography

- Recall:

$$x' = \frac{ax + by + c}{gx + hy + i}, \qquad y' = \frac{dx + ey + f}{gx + hy + i}$$

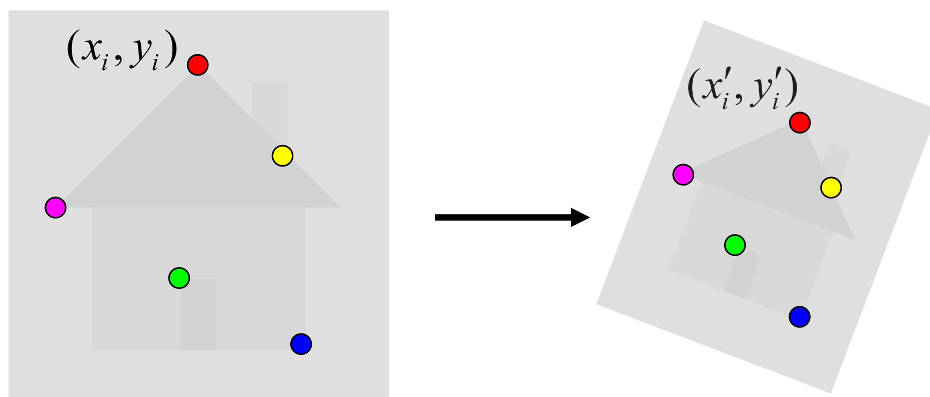


Alignment: Overview

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

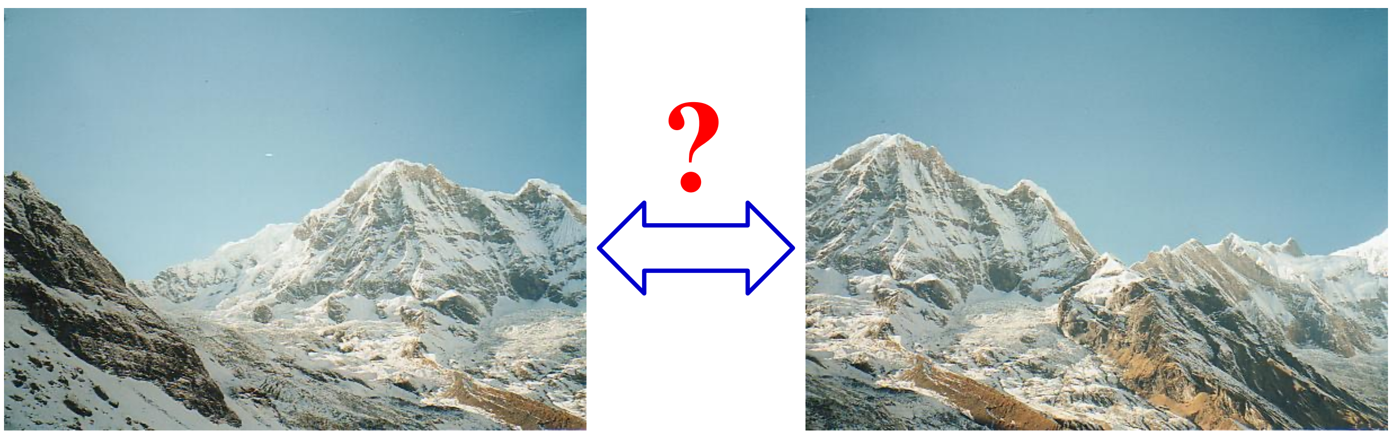
Robust feature-based alignment

- 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?



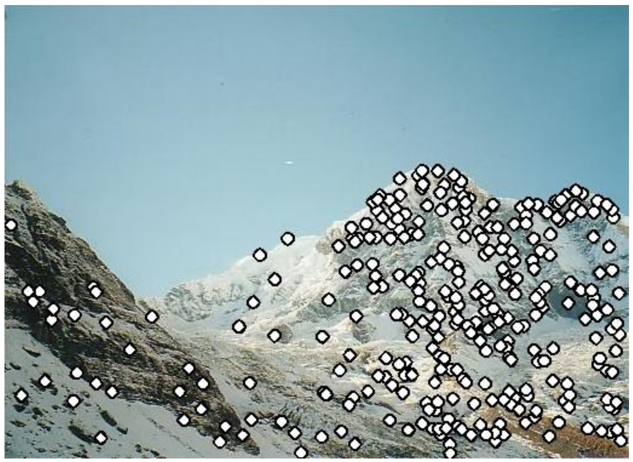
Robust feature-based alignment

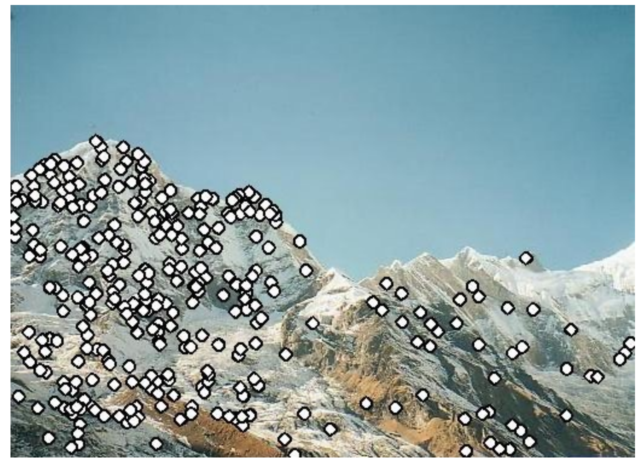
- So far, we've assumed that we are given a set of correspondences between the two images we want to align
- What if we don't know the correspondences?



Robust feature-based alignment

Robust feature-based alignment



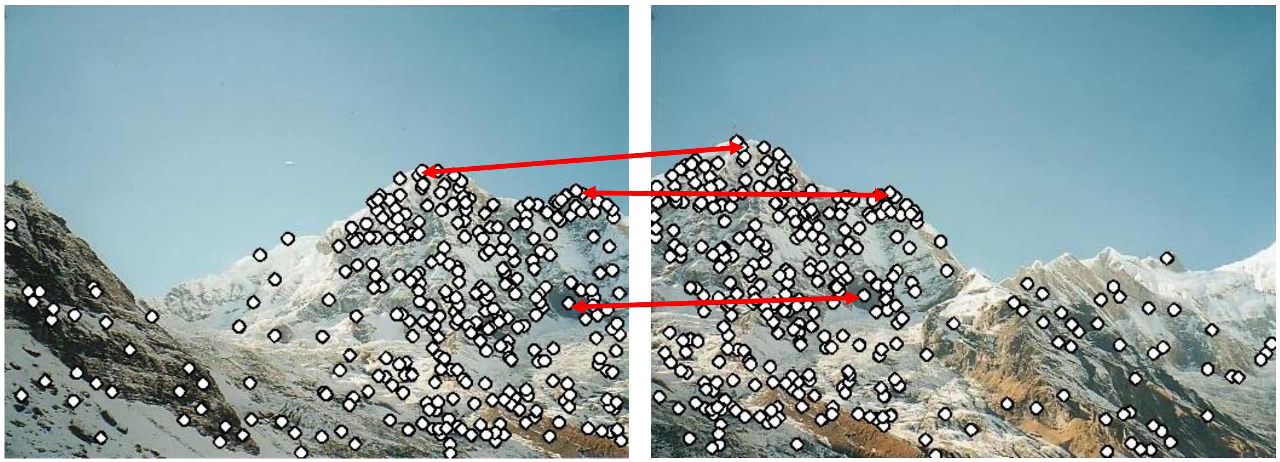


- Extract features

Robust feature-based alignment

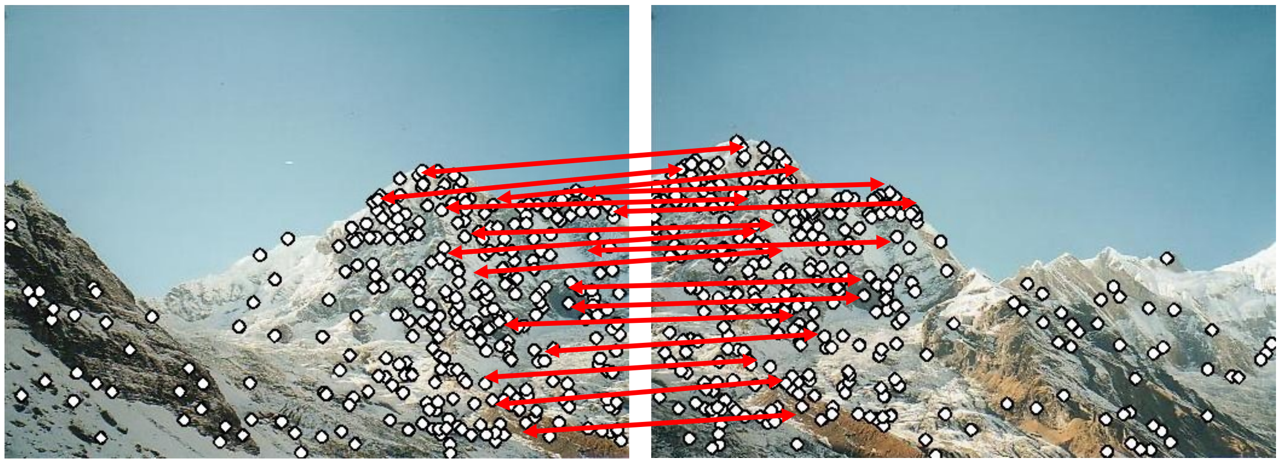
- Extract features
- Compute *putative matches*

Robust feature-based alignment



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

Robust feature-based alignment

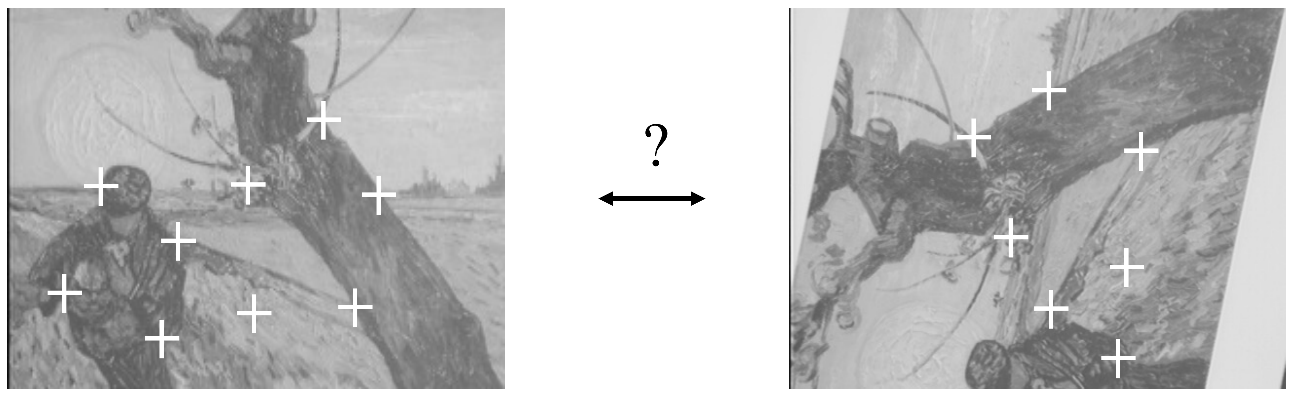


- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation T
  - *Verify* transformation (search for other matches consistent with T)

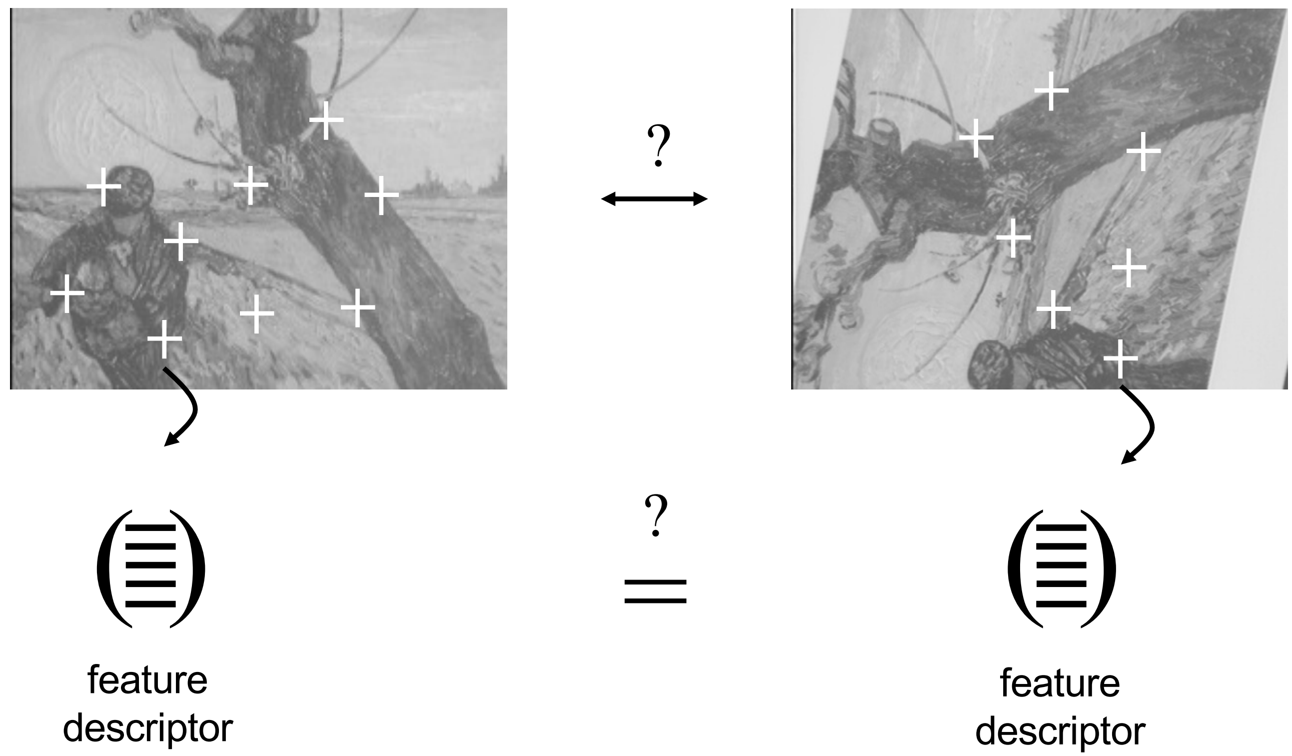
Robust feature-based alignment

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

Generating putative correspondences



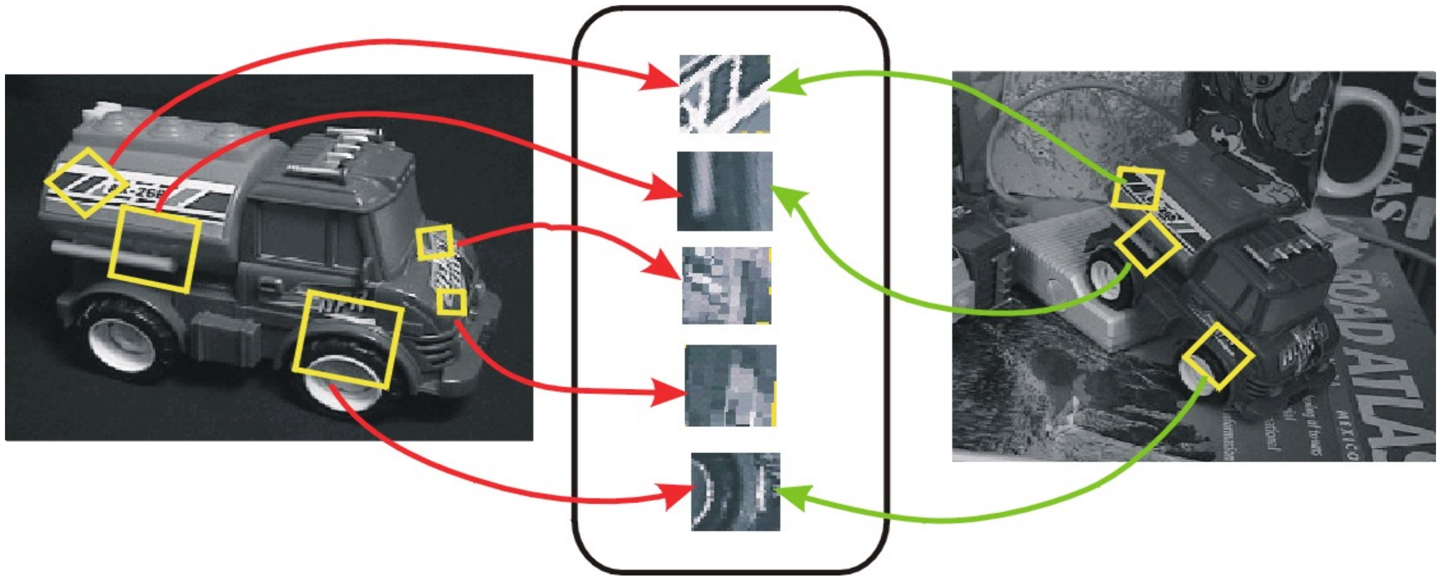
Generating putative correspondences



- 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):

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

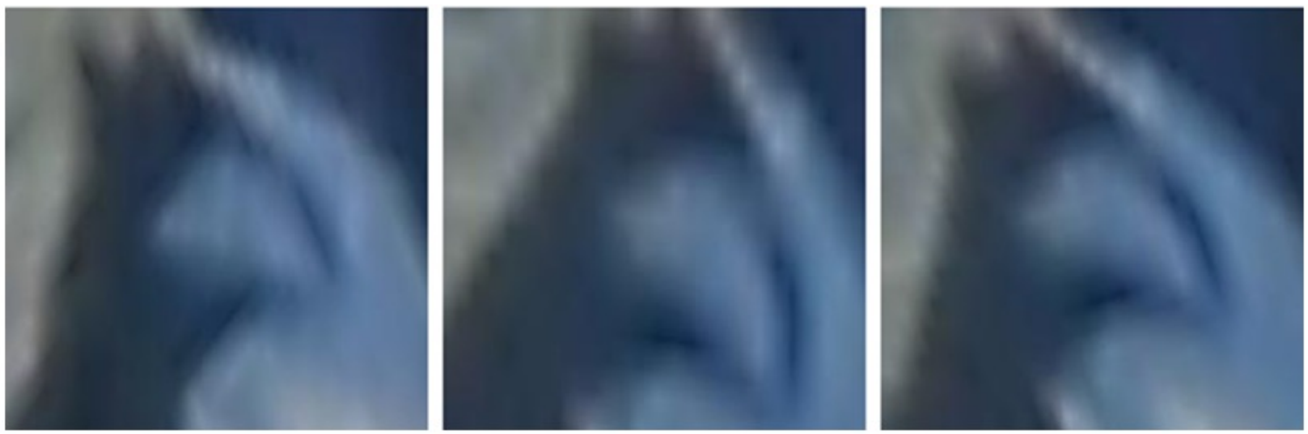
  - **Normalized correlation**: dot product between $\boldsymbol{u}$ and $\boldsymbol{v}$ normalized to have zero mean and unit norm:

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

  - Why would we prefer normalized correlation over SSD?

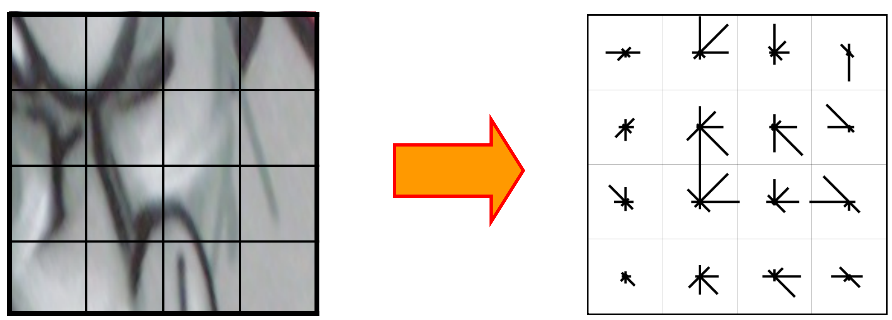
Disadvantage of intensity vectors as descriptors

- Small deformations can affect the matching score a lot



Feature descriptors: SIFT

- Descriptor computation:
  - Divide patch into $4{\times}4$ 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

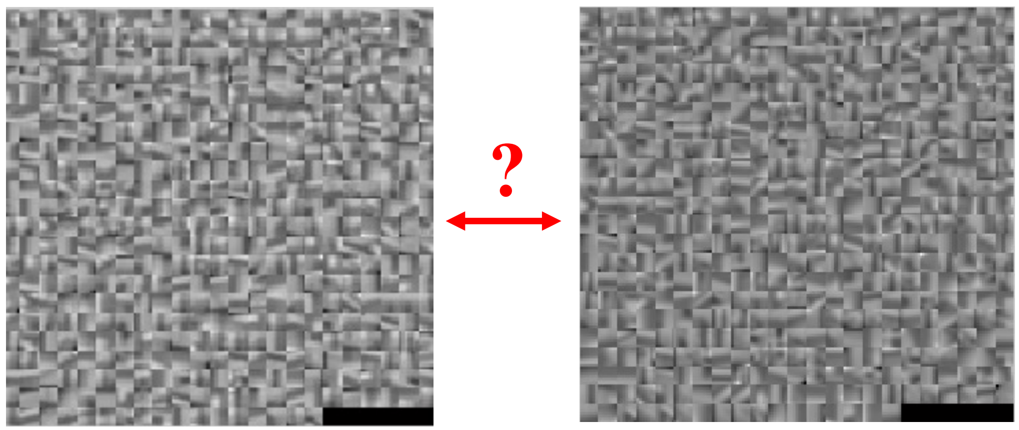
- Descriptor computation:
  - Divide patch into 4×4 sub-patches
  - Compute histogram of gradient orientations (8 reference angles) inside each sub-patch
  - Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions

- 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



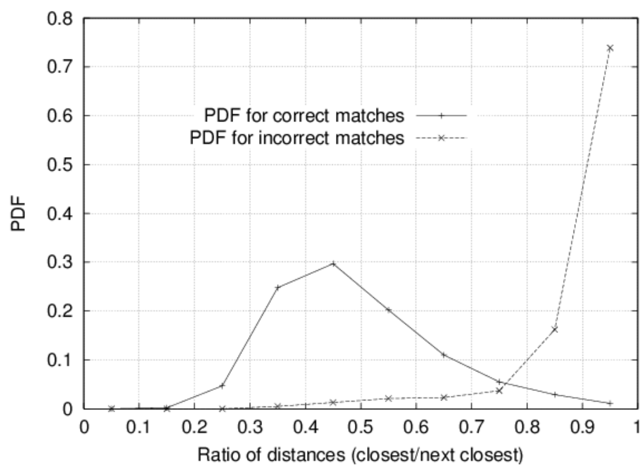
Rejection of ambiguous matches

- 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

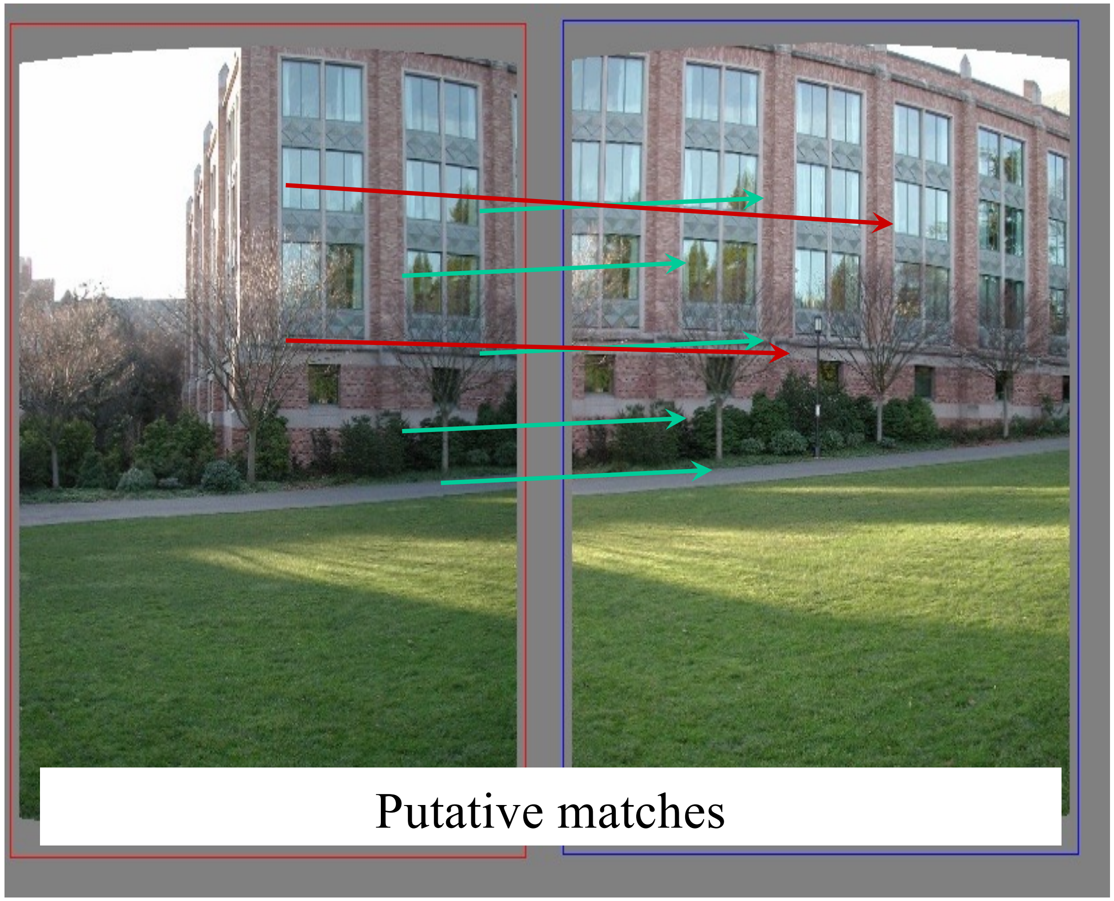
David G. Lowe. [Distinctive image features from scale-invariant keypoints](). *IJCV* 60 (2), pp. 91-110, 2004.

Robust alignment

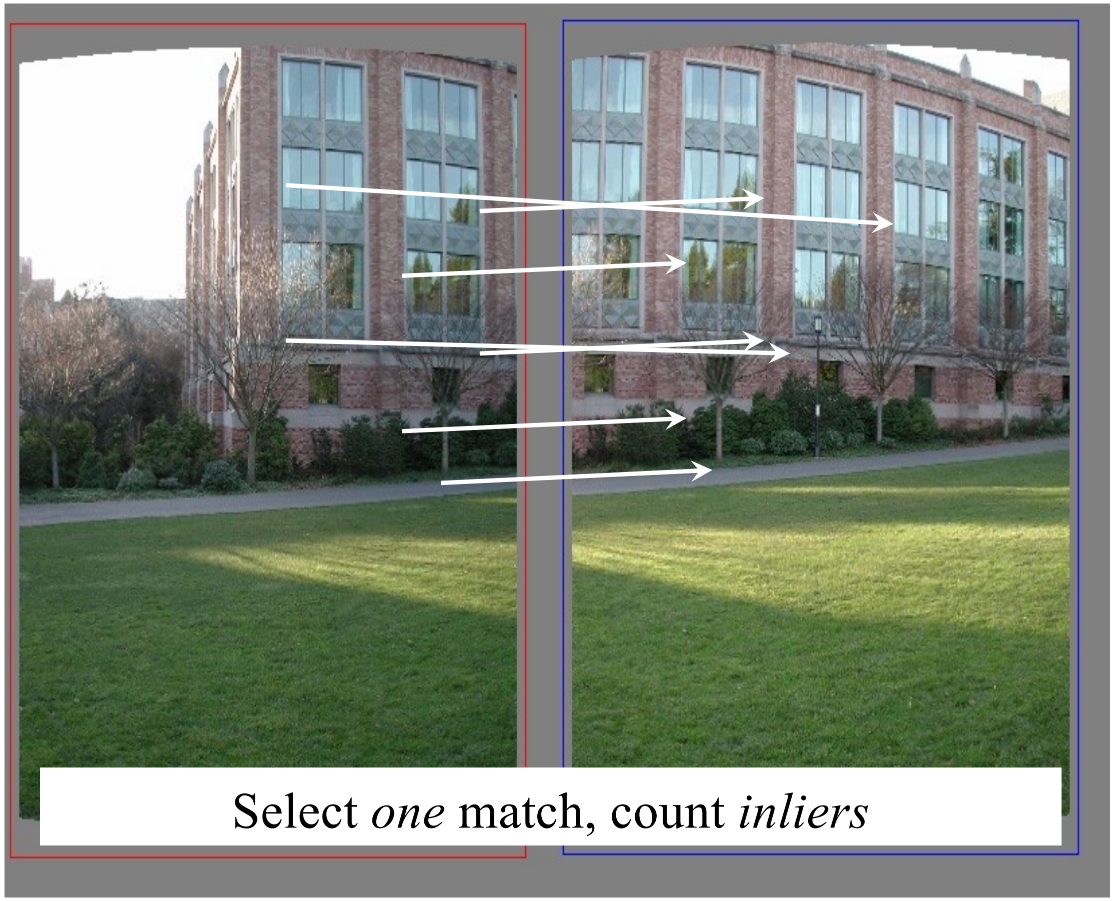
- Even after filtering out ambiguous matches, the set of putative matches still contains a very high percentage of outliers

- 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

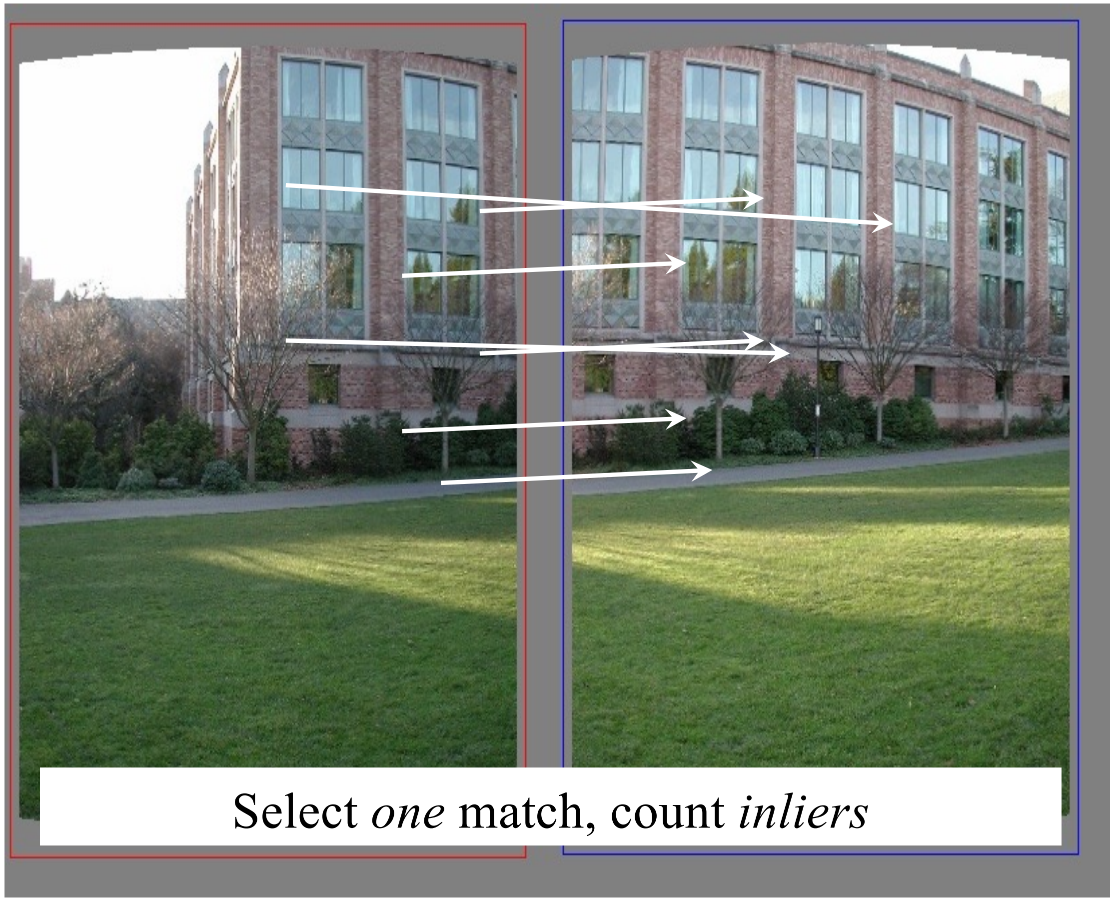
RANSAC example: Translation



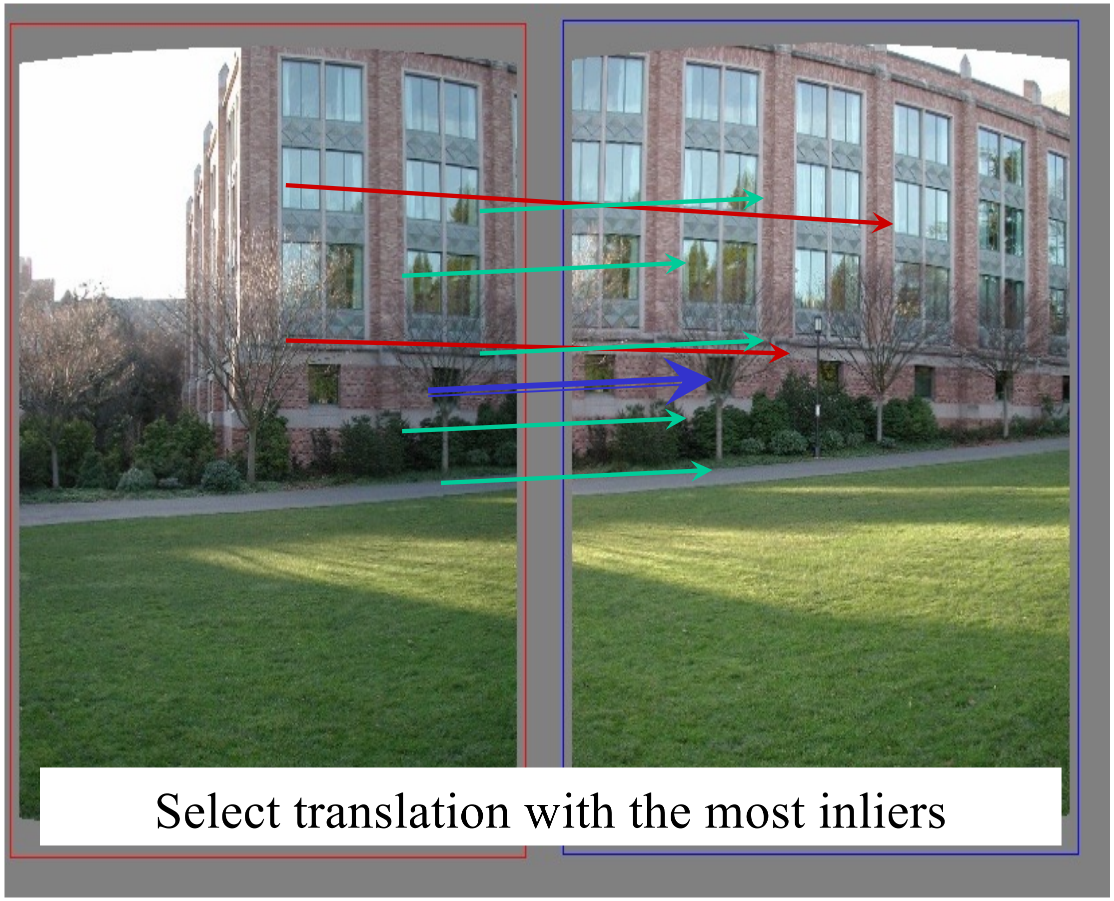
RANSAC example: Translation



RANSAC example: Translation

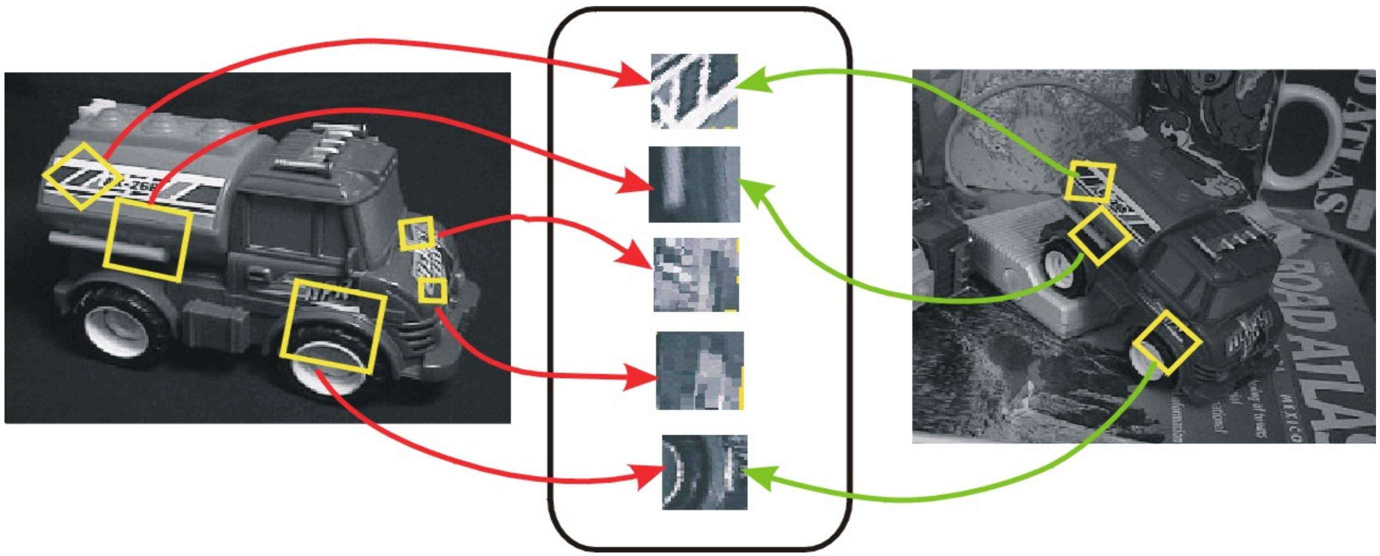


RANSAC example: Translation



Alternative to RANSAC: Voting

- A single SIFT match can vote for translation, rotation, and scale parameters of a transformation between two images
  - Votes are accumulated in a 4D coarsely discretized parameter space
  - Clusters of matches falling into the same bin undergo a more precise verification procedure



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

Other strategies if you don’t know correspondence

Iterated closest points:

Start with some transformation

Iterate:

- For each point in transformed source, find nearest neighbor in target
- Reestimate transformation using those correspondences
- Apply new transform to transformed source

Robust alignment

Robustness is a serious problem

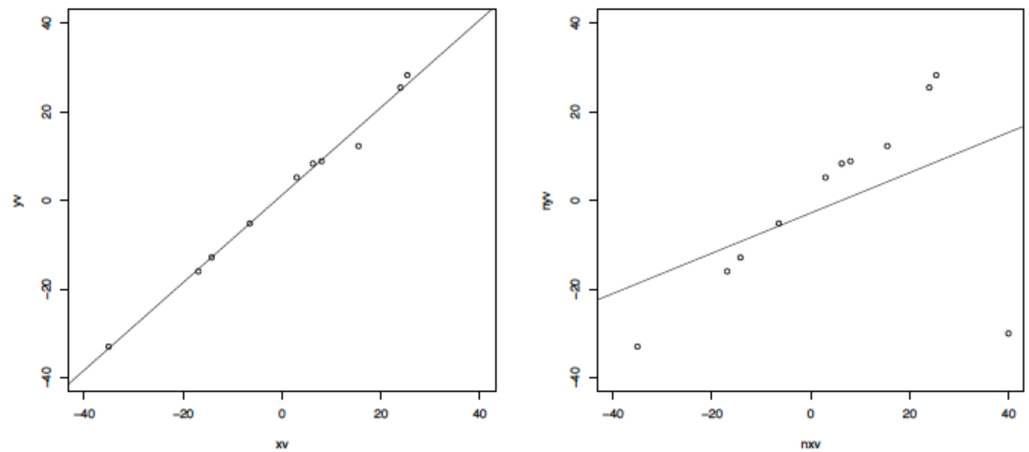


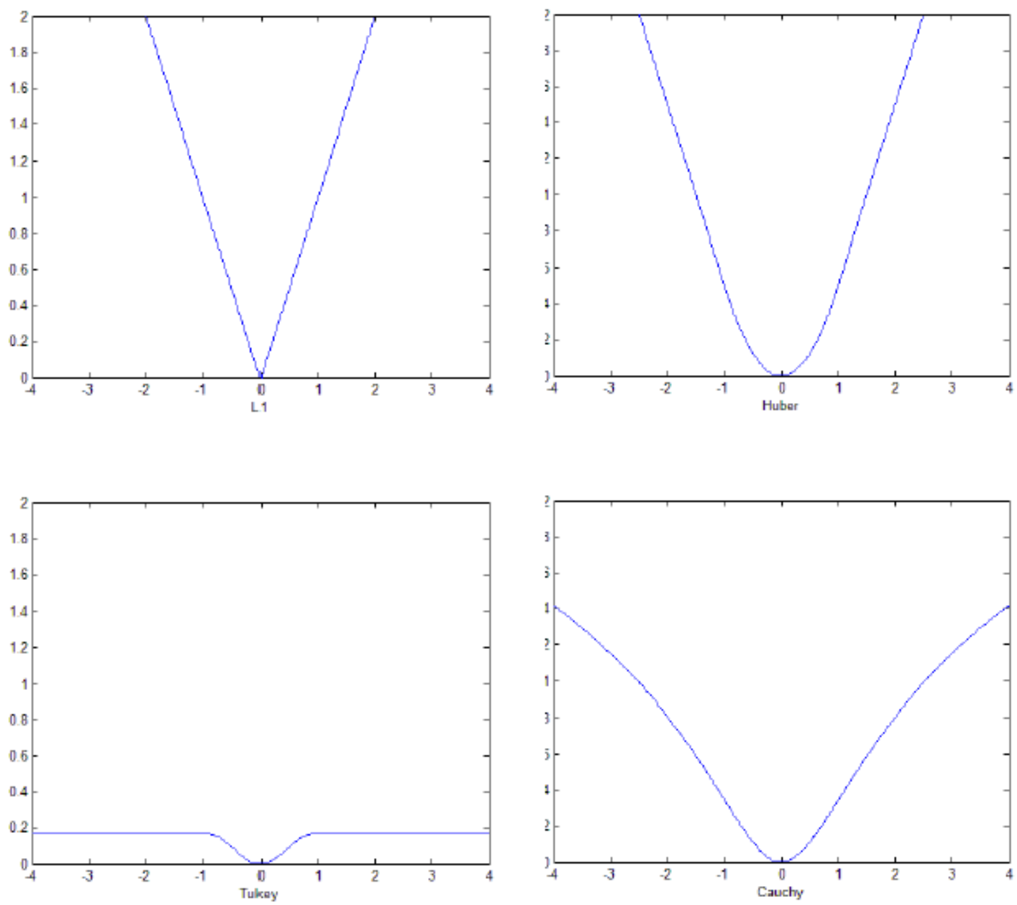
FIGURE 10.6: *On the* **left**, *a synthetic dataset with one independent and one explanatory variable, with the regression line plotted. Notice the line is close to the data points, and its predictions seem likely to be reliable. On the* **right**, *the result of adding a single outlying datapoint to that dataset. The regression line has changed significantly, because the regression line tries to minimize the sum of squared vertical distances between the data points and the line. Because the outlying datapoint is far from the line, the squared vertical distance to this point is enormous. The line has moved to reduce this distance, at the cost of making the other points further from the line.*

Key issue:

- Squaring a large number produces a huge number
- A few wildly mismatched points can throw off alignment
- Fixes:
  - remove matches with “large” distances
    - actually, quite good
    - but what happens if new such pairs emerge?
  - apply a robust loss
    - minimize

$$\sum_i \rho\left([\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}]^T [\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}]\right)$$

More robust loss functions



loss function	p(x)
L_1	$\lvert x \rvert$
$Huber \begin{cases} if\ \lvert x \rvert \le k \\ if\ \lvert x \rvert \ge k \end{cases}$	$\begin{cases} x^2/2 \\ k(\lvert x \rvert - \frac{k}{2}) \end{cases}$
$Tukey \begin{cases} if\ \lvert x \rvert \le k \\ if\ \lvert x \rvert > k \end{cases}$	$\begin{cases} k^2/6(1 - \left(1 - \left(\frac{x}{c}\right)\right. \\ k^2/6 \end{cases}$
$Cauchy$	$\frac{k^2}{2}\log(1 + (x/k)^2)$

An affine transformation, harder linear algebra

In the simplest case, the correspondence is known – perhaps $\mathcal{Y}$ consists of beacons and $\mathcal{X}$ of observations – and the only noise is Gaussian (so $N = M$). We will assume the noise is isotropic, which is by far the most usual case. Once you have followed this derivation, you will find it easy to incorporate a known covariance matrix. We have

$$\mathbf{x}_i = \mathcal{M}\mathbf{y}_i + \mathbf{t} + \xi_i \tag{12.1}$$

where ξ_i is the value of a normal random variable with mean $\mathbf{0}$ and covariance matrix $\Sigma = \sigma^2\mathcal{I}$. A natural procedure to estimate $\mathcal{M}$ and $\mathbf{t}$ is to maximize the likelihood of the noise. Because it will be useful later, we assume that there is a weight w_i for each pair, so the negative log-likelihood we must minimize is proportional to

$$\sum_i w_i \left(\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right)^T \left(\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right) \tag{12.2}$$

Weights (assume known)

An affine transformation, harder linear algebra

The translation is easy to estimate

(difference between weighted centers of gravity)

lem). The gradient of this cost with respect to $\mathbf{t}$ is

$$-2\sum_i w_i\left(\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right) \tag{12.3}$$

which vanishes at the solution. In turn, if $\sum_i w_i \mathbf{x}_i = \sum_i w_i \mathcal{M}\mathbf{y}_i$, $\mathbf{t} = \mathbf{0}$. One straightforward way to achieve this is to ensure that both the observations and the reference points have a center of gravity at the origins. Write

$$\mathbf{c}_x = \frac{\sum_i w_i \mathbf{x}_i}{\sum_i w_i} \tag{12.4}$$

for the center of gravity of the observations (etc.) Now form

$$\mathbf{u}_i = \mathbf{x}_i - \mathbf{c}_x \text{ and } \mathbf{v}_i = \mathbf{y}_i - \mathbf{c}_y \tag{12.5}$$

An affine transformation, harder linear algebra

We obtain $\mathcal{M}$ by minimizing

$$\sum_i w_i \left(\mathbf{u}_i - \mathcal{M}\mathbf{v}_i\right)^T \left(\mathbf{u}_i - \mathcal{M}\mathbf{v}_i\right). \tag{12.6}$$

Now write $\mathcal{W} = \mathsf{diag}\left([w_1, \ldots, w_N]\right)$, $\mathcal{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ (and so on). You should check that the objective can be rewritten as

$$\mathsf{Tr}\left(\mathcal{W}(\mathcal{U} - \mathcal{M}\mathcal{V})^T(\mathcal{U} - \mathcal{M}\mathcal{V})\right). \tag{12.7}$$

Now the trace is linear; $\mathcal{U}^T\mathcal{U}$ is constant; and we can rotate matrices through the trace (Section 34.2). This means the cost is equivalent to

$$\mathsf{Tr}\left(-2\mathcal{M}\mathcal{V}\mathcal{U}^T + \mathcal{M}^T\mathcal{M}\mathcal{V}\mathcal{V}^T\right) \tag{12.8}$$

which will be minimized when

Target points **Weights** **Source points**

$$\mathcal{M}\mathcal{V}\mathcal{W}\mathcal{V}^T = \mathcal{V}\mathcal{W}\mathcal{U}^T \tag{12.9}$$

What we want

(which you should check). Many readers will recognize a least squares solution here. The trace isn't necessary here, but it's helpful to see an example using the trace, because it will be important in the next case.

Robust alignment

Iterative reweighted least squares:

iterate:

- Align using weights
- Adjust weights

But how?

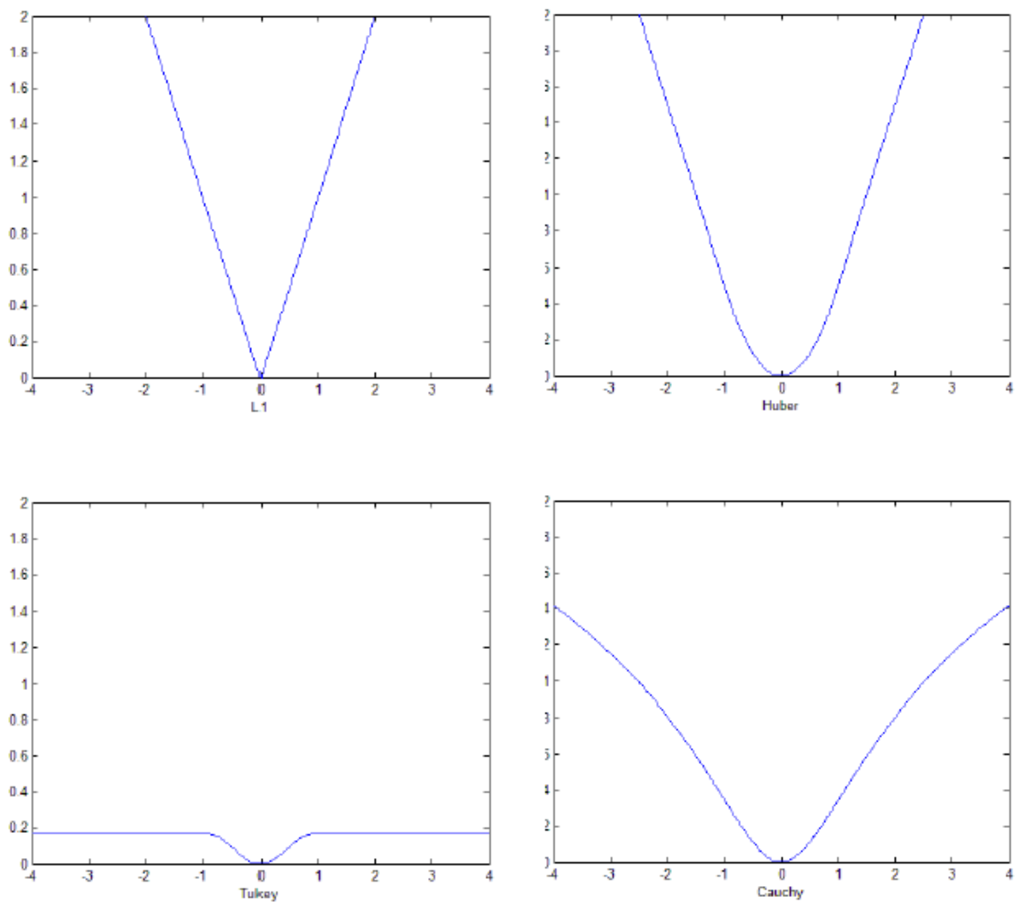
Using a robust loss

minimize

But how?

$$\sum_i \rho\left(\left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right]^T \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right]\right)$$

More robust loss functions



loss function	p(x)
L_1	$\|x\|$
$Huber\begin{cases} if\ \|x\| \le k \\ if\ \|x\| \ge k \end{cases}$	$\begin{cases} x^2/2 \\ k(\|x\| - \frac{k}{2}) \end{cases}$
$Tukey\begin{cases} if\ \|x\| \le k \\ if\ \|x\| > k \end{cases}$	$\begin{cases} k^2/6(1 - \left(1 - \left(\frac{x}{c}\right)\right. \\ k^2/6 \end{cases}$
$Cauchy$	$\frac{k^2}{2}\log(1 + (x/k)^2)$

Analogy

Weighted least squares minimizes:

$$\sum_i w_i \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right]^T \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right])$$

So at a solution

$$\sum_i w_i \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right] = 0$$

Robust loss minimizes:

$$\sum_i \rho(\left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right]^T \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right])$$

So at a solution

$$\sum_i \rho' \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right] = 0$$

Analogy

Weighted least squares minimizes:

$$\sum_i w_i \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right]^T \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right])$$

So at a solution

$$\sum_i w_i \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right] = 0$$

Robust loss minimizes:

$$\sum_i \rho(\left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right]^T \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right])$$

So at a solution

$$\sum_i \rho' \left[\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right] = 0$$

Idea

Iterativelyreweighted least squares:

start with weights=1

iterate:

- Align using weights
- Set weights to ρ'_i

Any solution to a robust loss will certainly be a stationary point of this iteration

Example:

Robust loss is absolute value, so

$$\rho([\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}]^T [\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}]) = \mathrm{abs}([\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}])$$

Write s for squared distance

$$\rho(s) = \sqrt{s}$$

We have

$$\rho' = \frac{1}{2\sqrt{s}}$$

What happens in iteration?

Iterate

compute weighted transformation

If transformed source point is close to target, weight pair up

If transformed source point is far from target, weight pair down

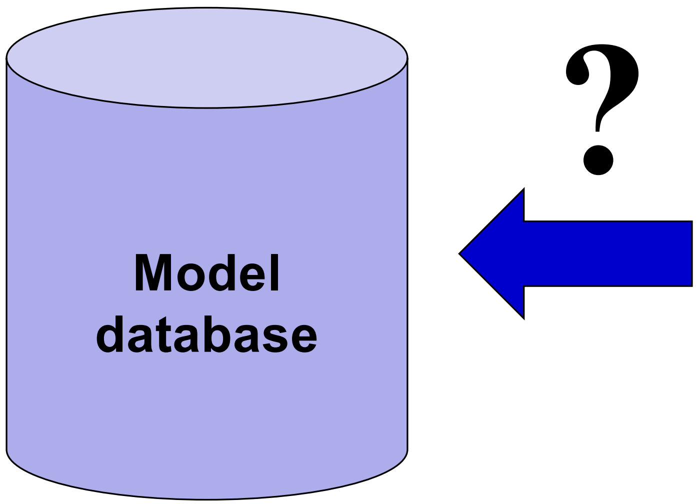
If transformed target lies on source, divide by zero!

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

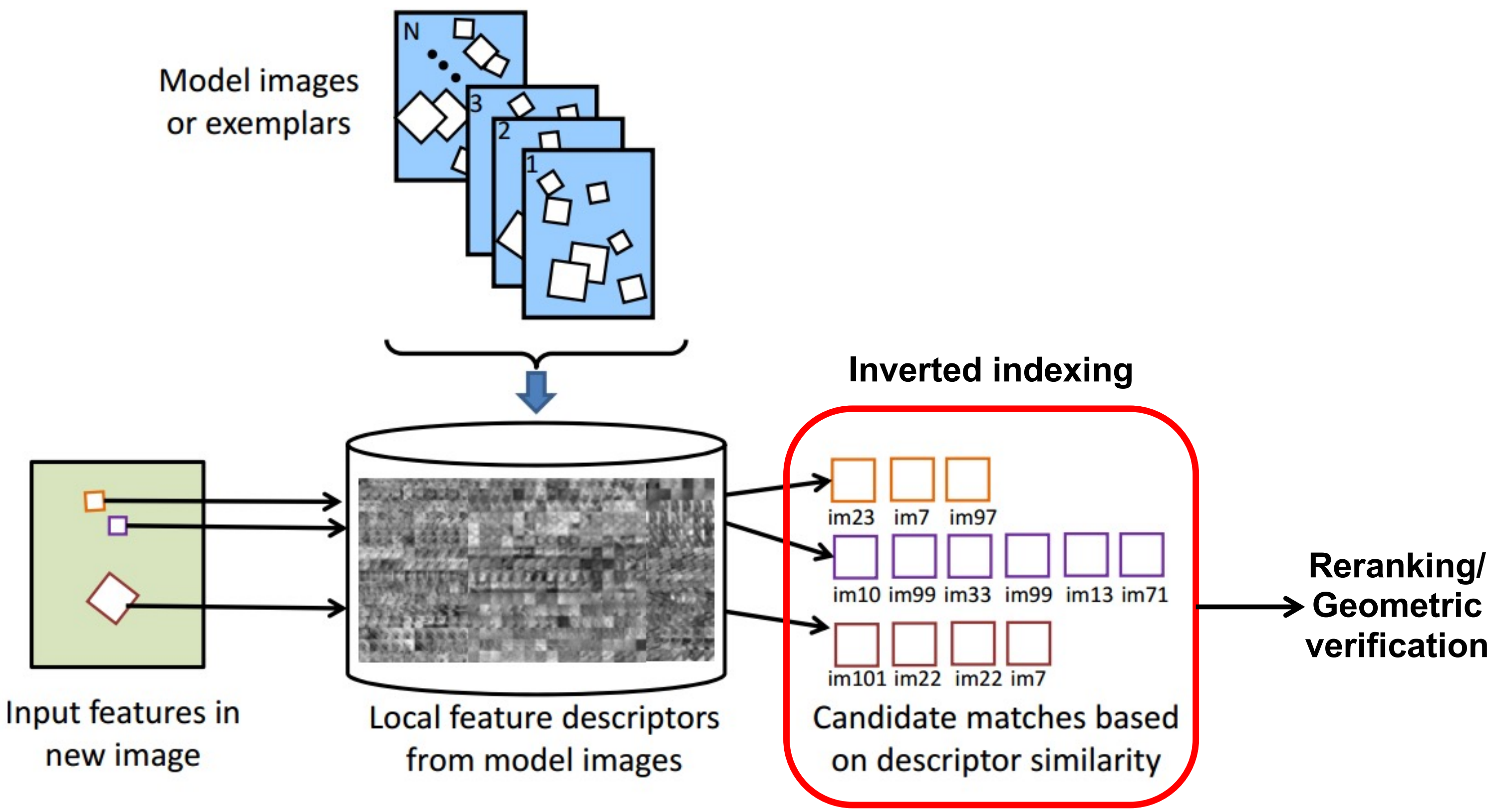
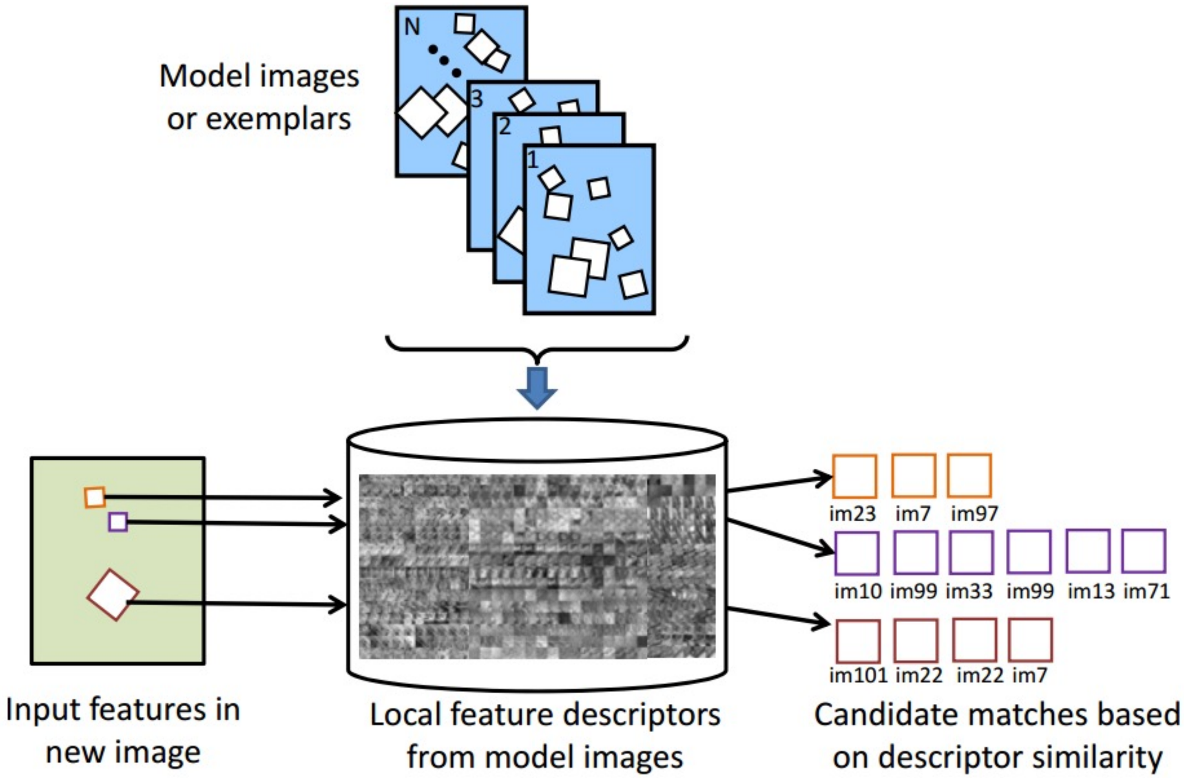


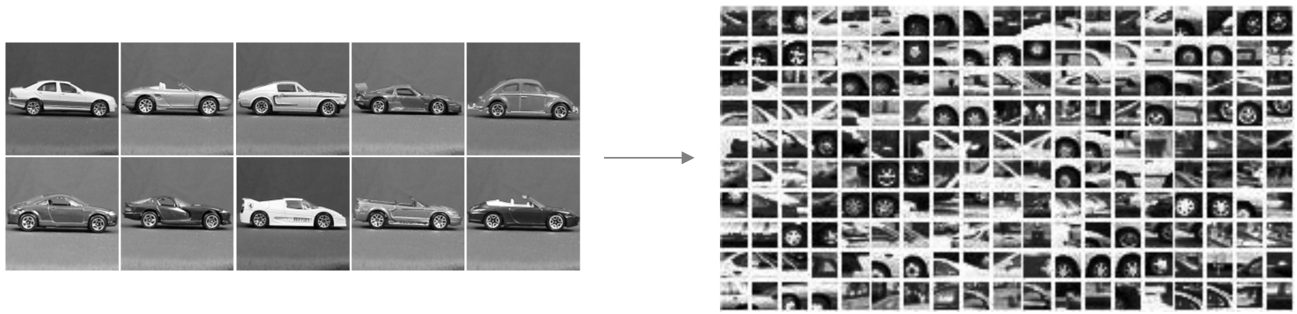
Figure from: Kristen Grauman and Bastian Leibe, [Visual Object Recognition](), Synthesis Lectures on Artificial Intelligence and Machine Learning, April 2011, Vol. 5, No. 2, pp. 1-181

How to do the indexing?

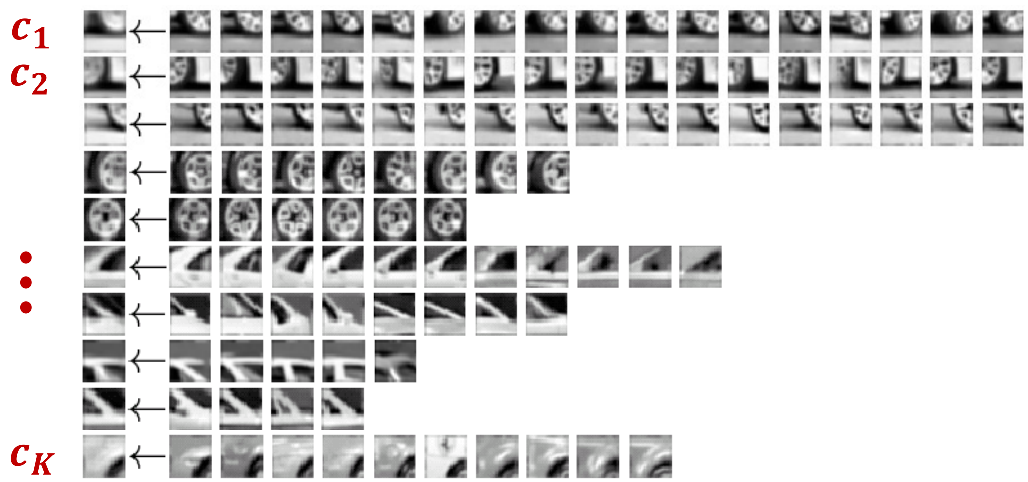


- Idea: find a set of *visual codewords* to which descriptors can be *quantized*

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:

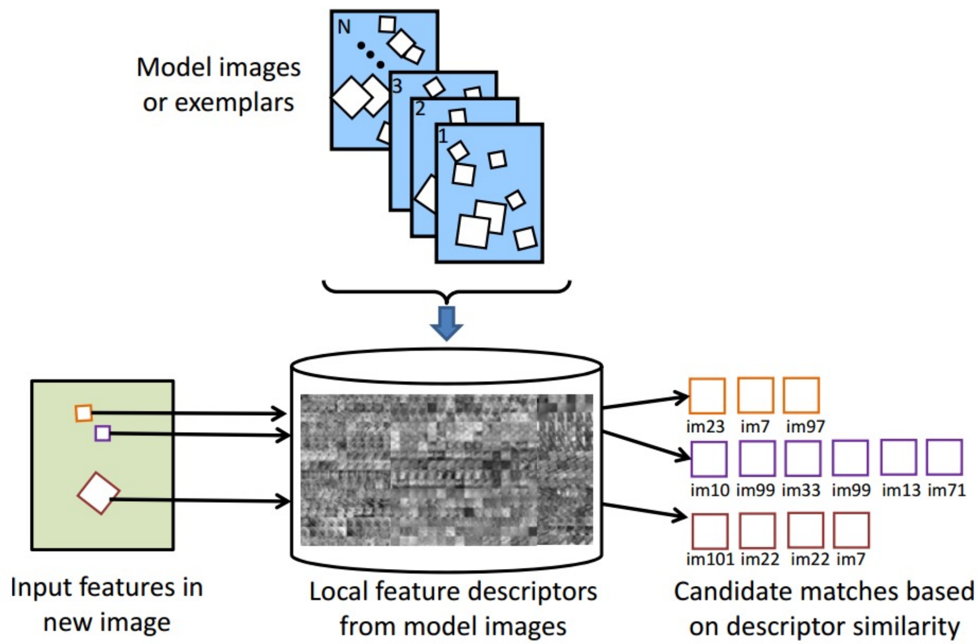
$$\sum_i \sum_k a_{ik} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2$$

- 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

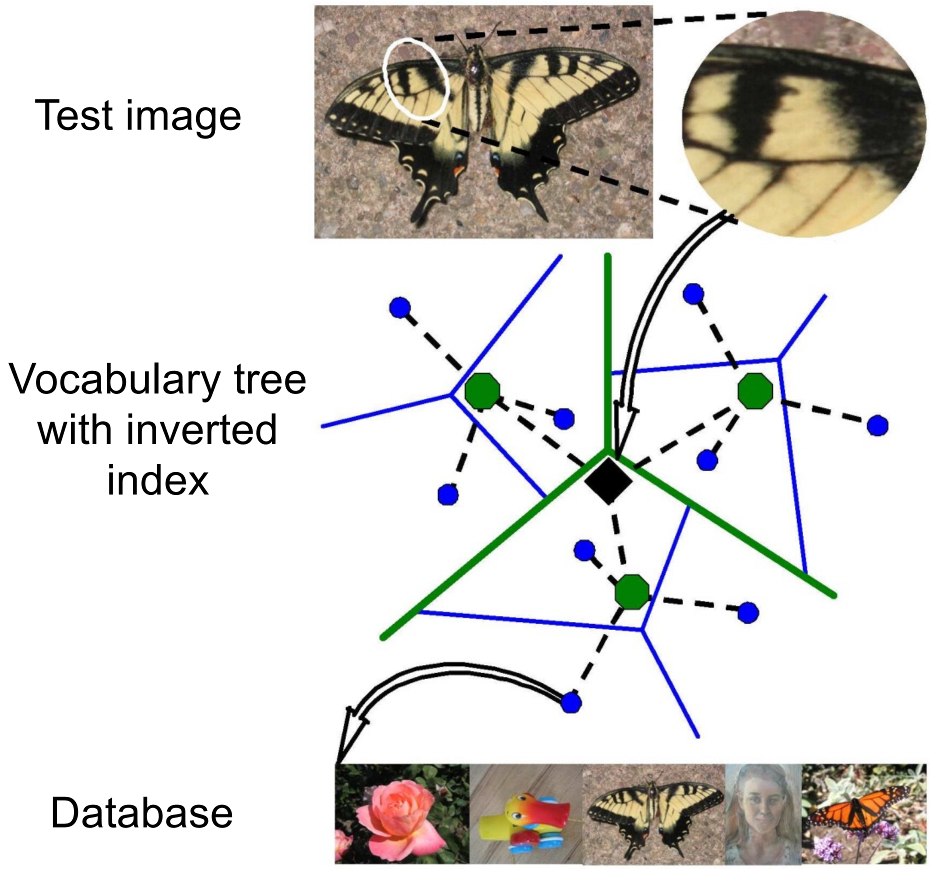
[Source]

How to do the indexing?

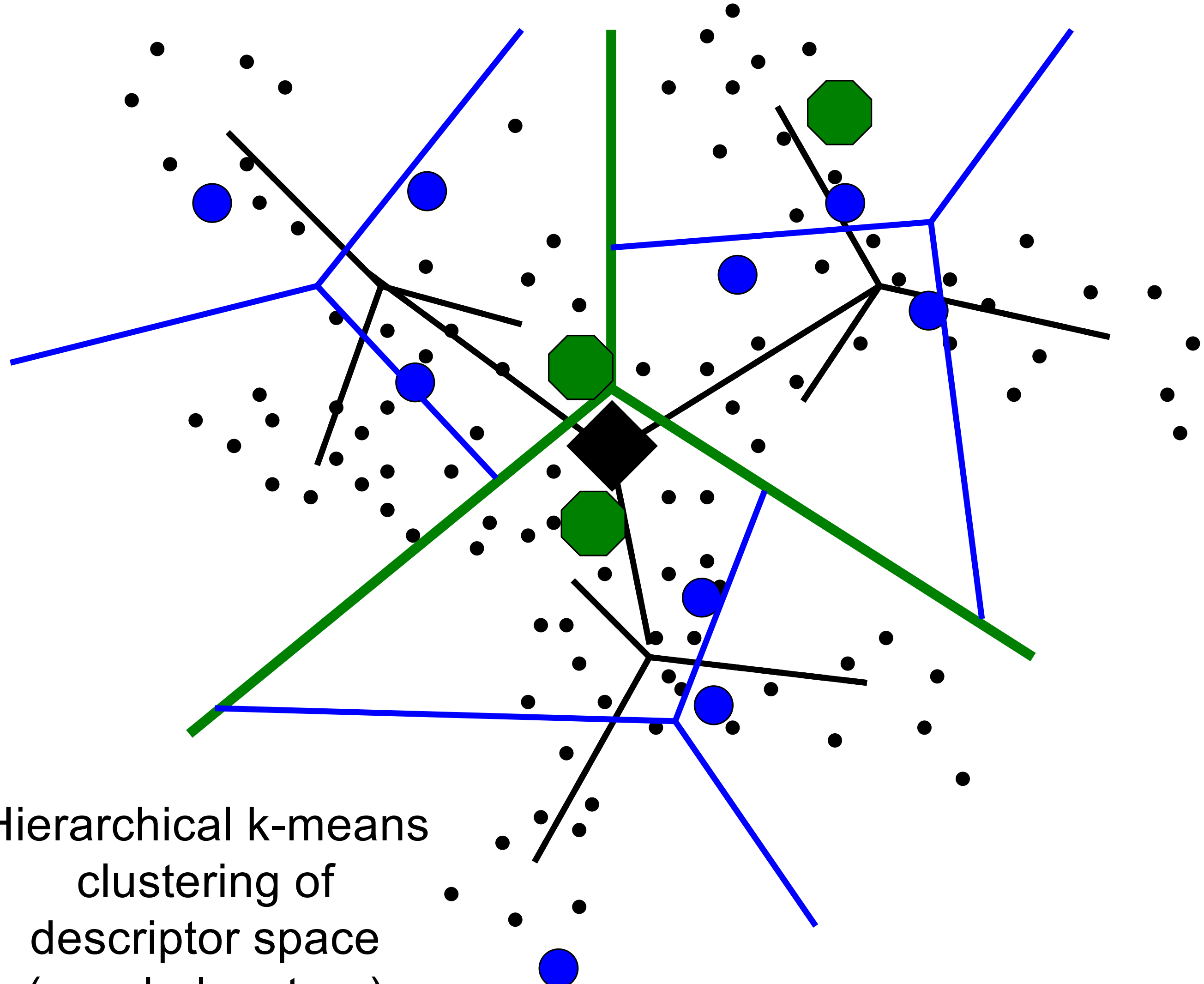


- Cluster descriptors in the database to form codebook
- At query time, quantize descriptors in query image to nearest codevectors
- Problem solved?

Efficient indexing technique: Vocabulary trees

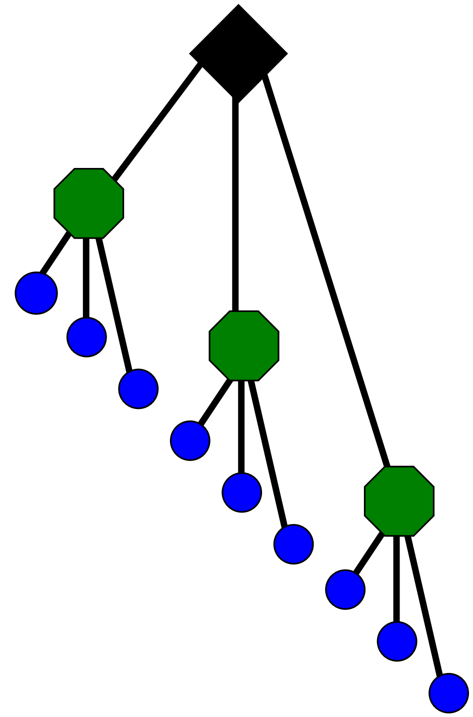


D. Nistér and H. Stewénius, [Scalable Recognition with a Vocabulary Tree](), CVPR 2006



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

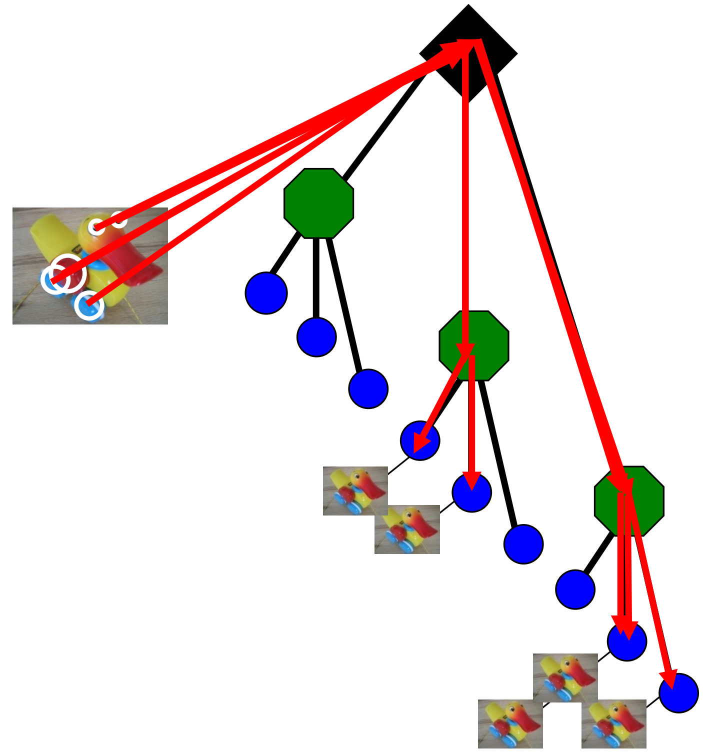
Slide credit: D. Nister



Vocabulary tree/inverted index

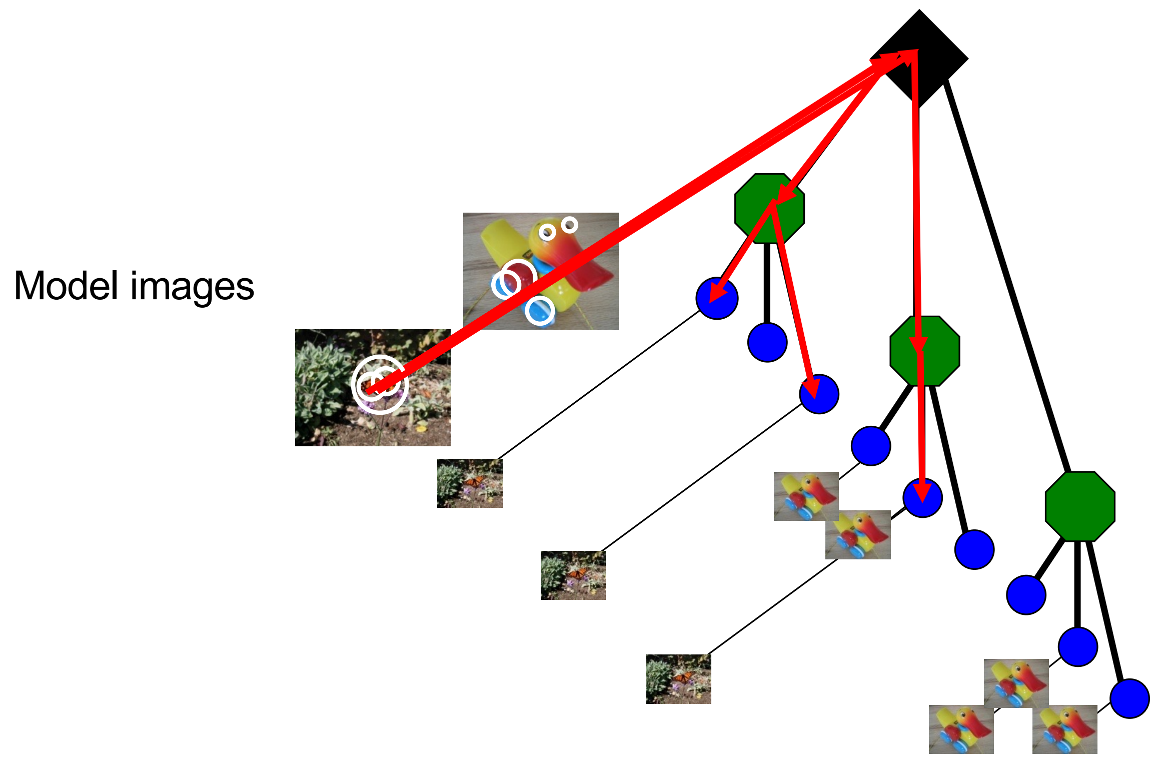
Slide credit: D. Nister

Model images



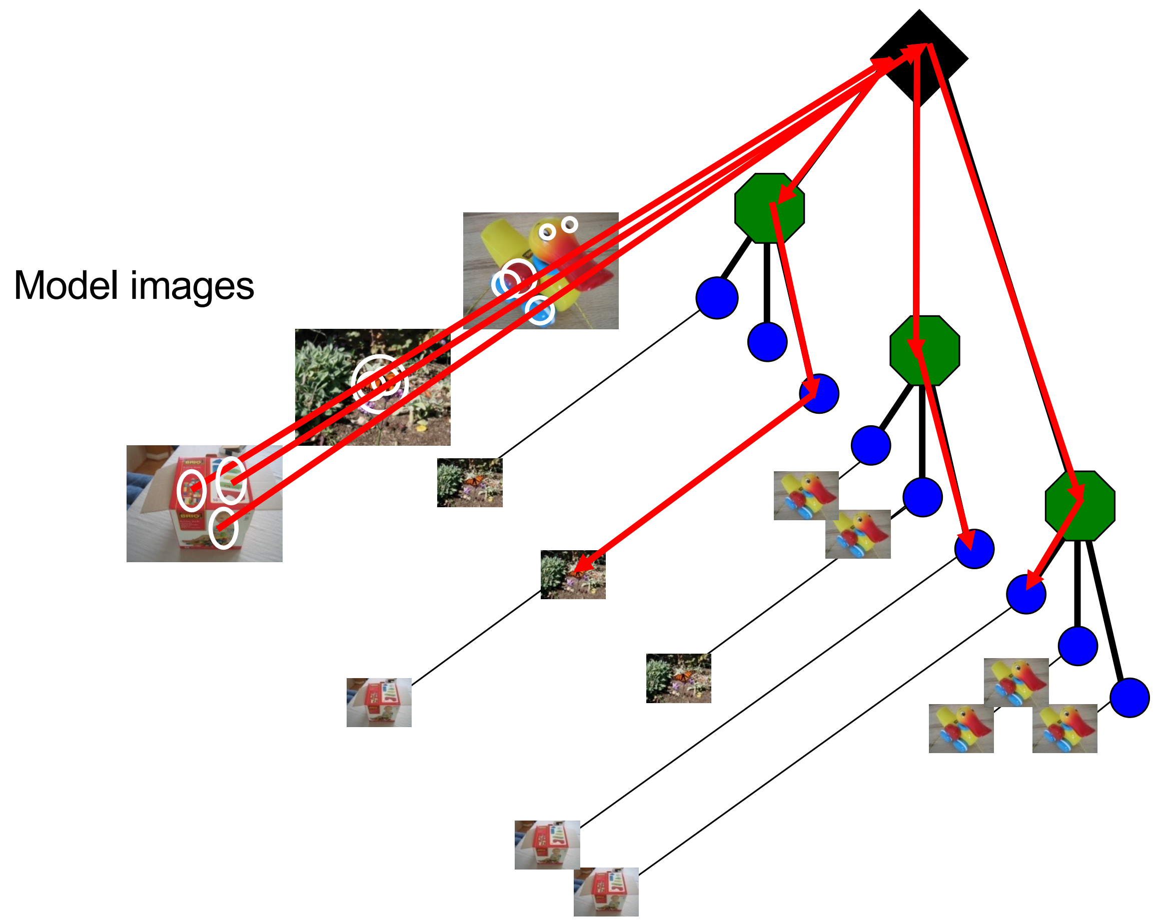
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



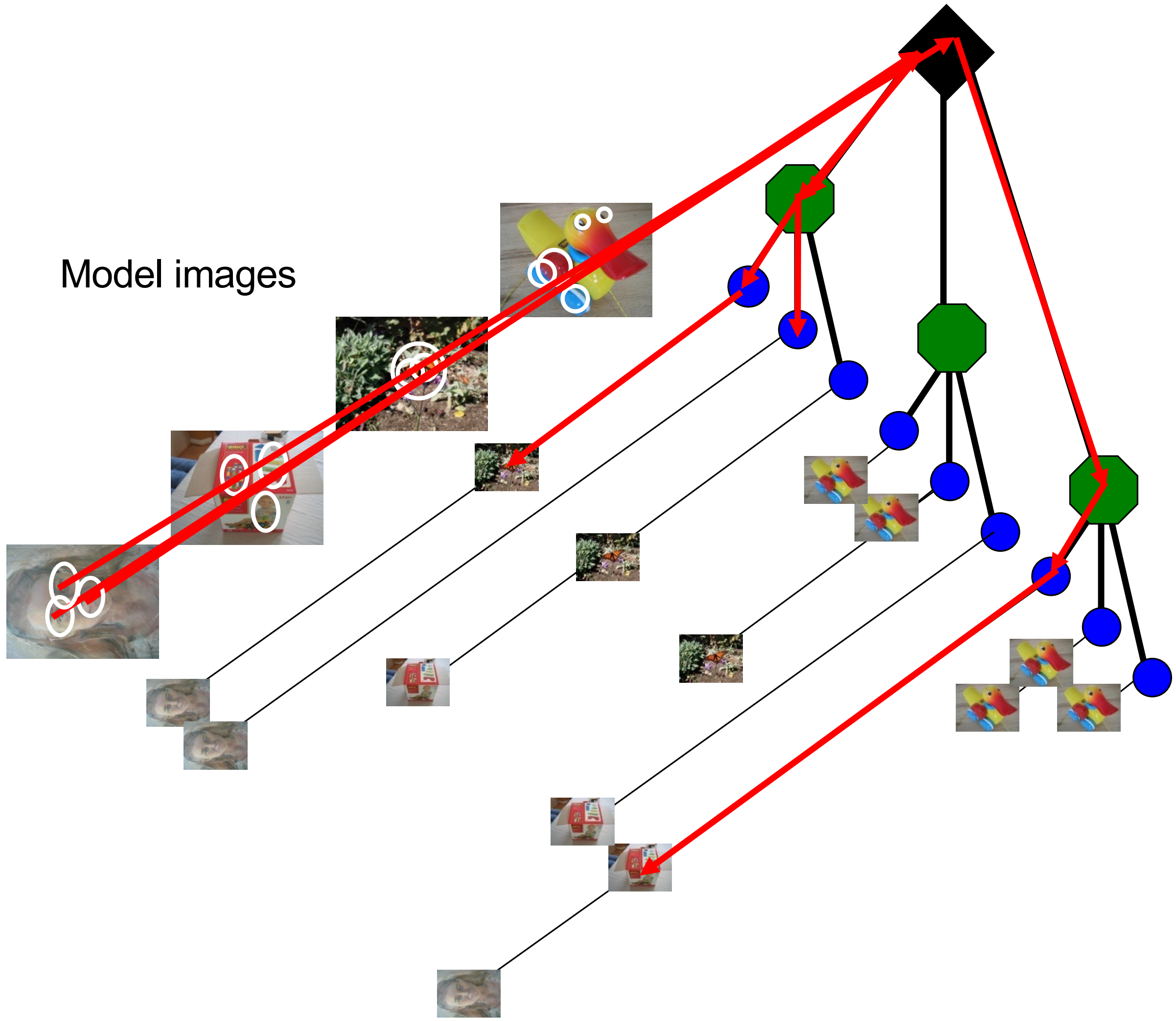
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



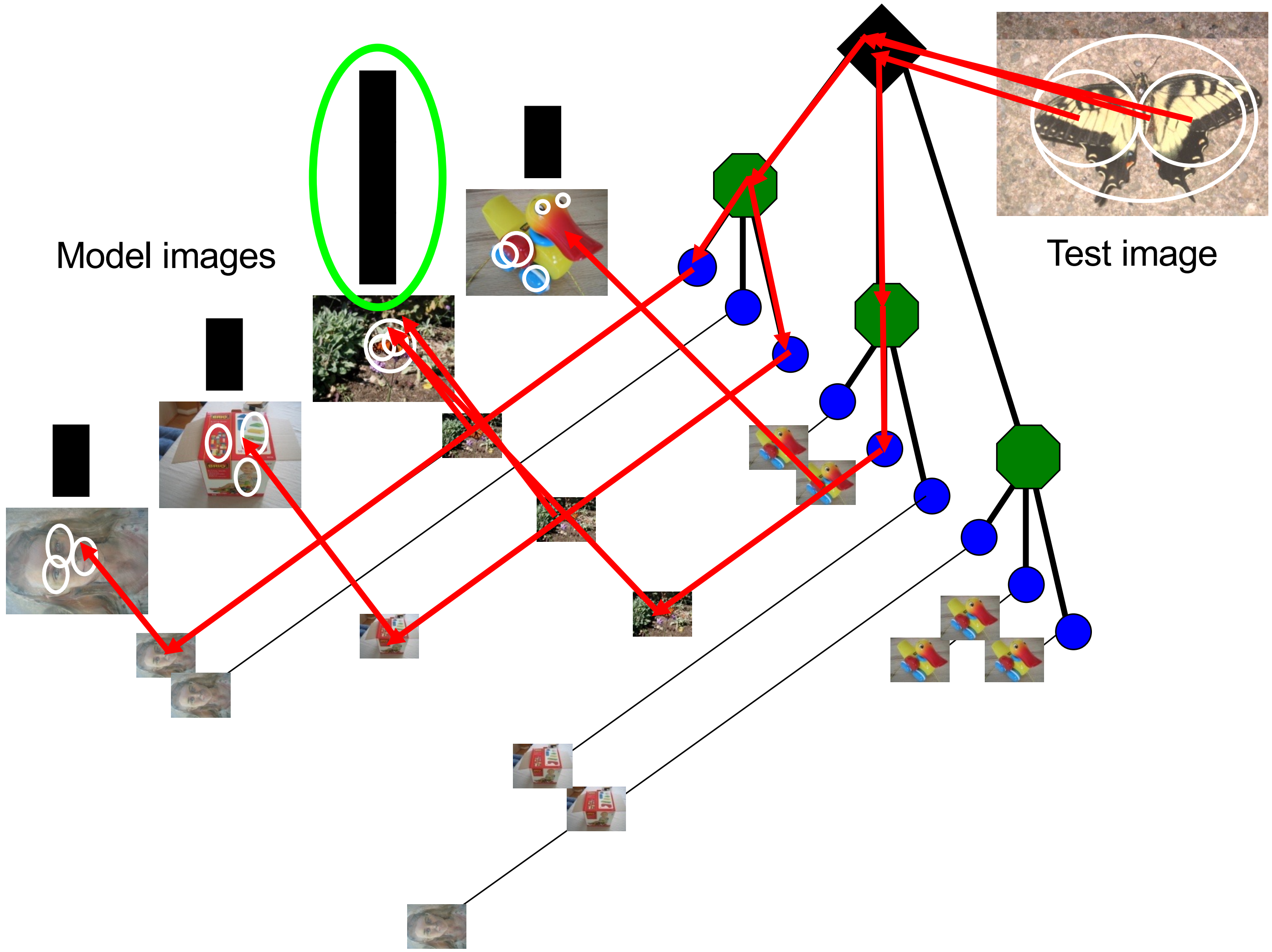
Populating the vocabulary tree/inverted index

Slide credit: D. Nister



Populating the vocabulary tree/inverted index

Slide credit: D. Nister



Looking up a test image

Slide credit: D. Nister