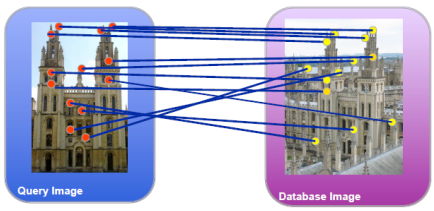
Geometric verification of matching

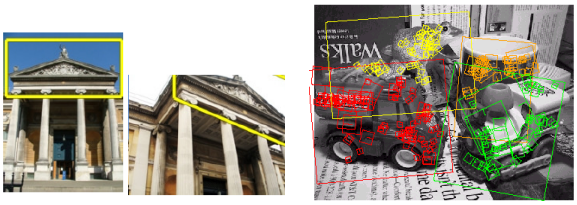
The correspondence problem

- The correspondence problem tries to figure out which parts of an image correspond to which parts of another image, after the camera has moved, time has elapsed, and/or the objects have moved around.
- Given two or more images of the same 3D scene, taken from different points of view, at different times, and with objects in the scene in general motion relative to the camera, the correspondence problem is:
  - to find a set of points in one image which can be identified as the same points in another image i.e. verify if they belong to a consistent configuration
- Image content is transformed into local features that are invariant to translation, rotation, and scale and the correspondence is assessed by checking if the layout of a subset of features is similar in the two images.



RANSAC RANdom SAmple Consensus

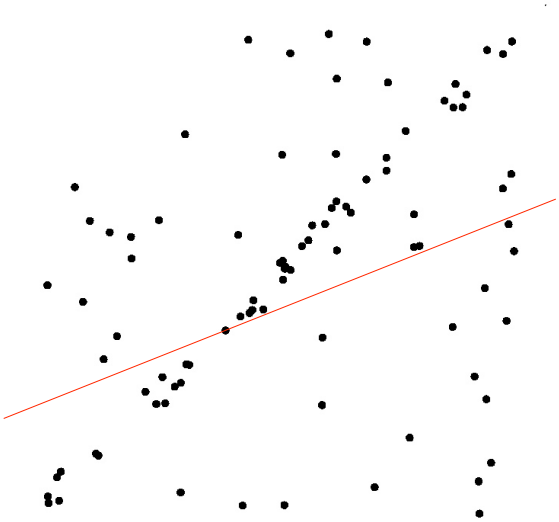
- A classical solution to the correspondence problem is the RANSAC algorithm [Fischler81]. RANSAC permits the estimation of parameters of a mathematical model by random sampling.
- The basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, and "outliers" which are data that do not fit the model. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data:
  - It assumes that objects are planar, (valid for many structures on buildings and man made objects; sufficient for small viewpoint variations on 3D objects)
  - It is non-deterministic in that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed.
  - The main disadvantage is that no upper bound exists on the time required to compute the parameters.



The RANSAC algorithm

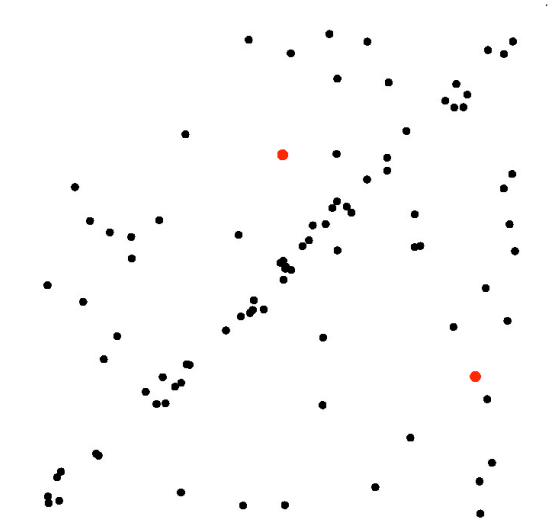
- Input:
  - a set of observed data values;
  - a parameterized model which can explain or be fitted to the observations;
  - confidence parameters.
- Generate k (a predetermined number) model hypotheses, each of them is computed using a minimal subset m of points
- For each model hypothesis
  - Draw a sample of m points from data at random
  - The parameters of the model are reconstructed from the set of points
  - Compute the residuals with respect to all data points. Points with residuals less than some threshold t are classified as *hypothetical inliers*
  - The estimated model is reasonably good if sufficiently many points have been classified as hypothetical inliers.
  - The model is reestimated from all hypothetical inliers (it was only been estimated from the initial set of hypothetical inliers).
  - The model is evaluated by estimating the error of the inliers relative to the model.
- end
- At each iteration a model is produced that either is rejected because too few points are classified as inliers or a is a refined model with a corresponding error measure. The refined model is accepted if its error is lower than the last saved model.

Example: fitting a line

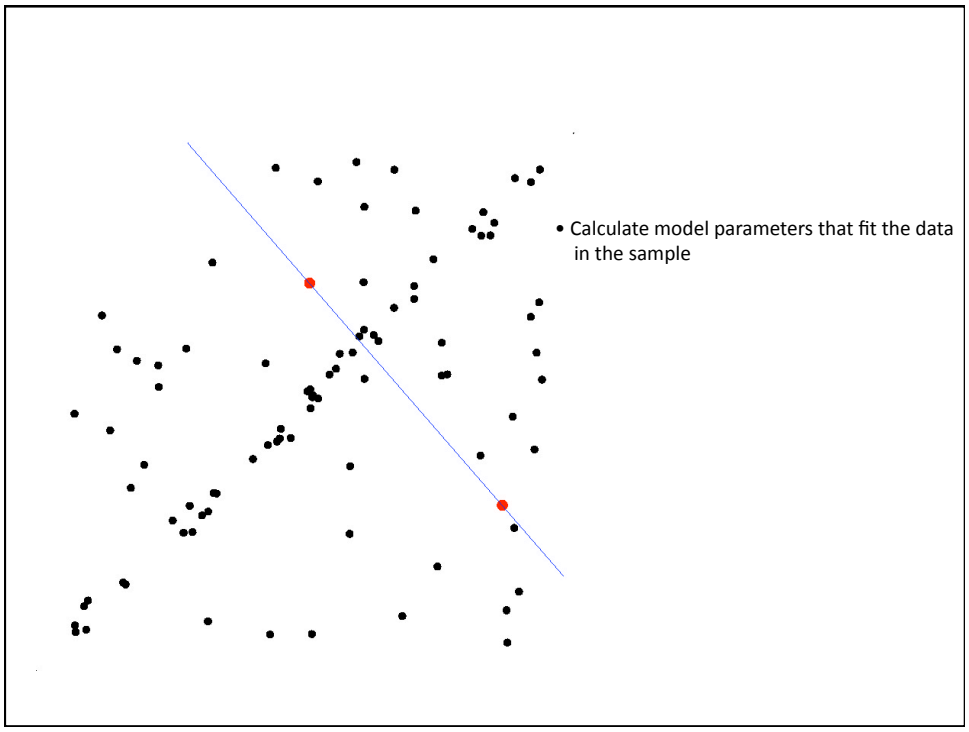


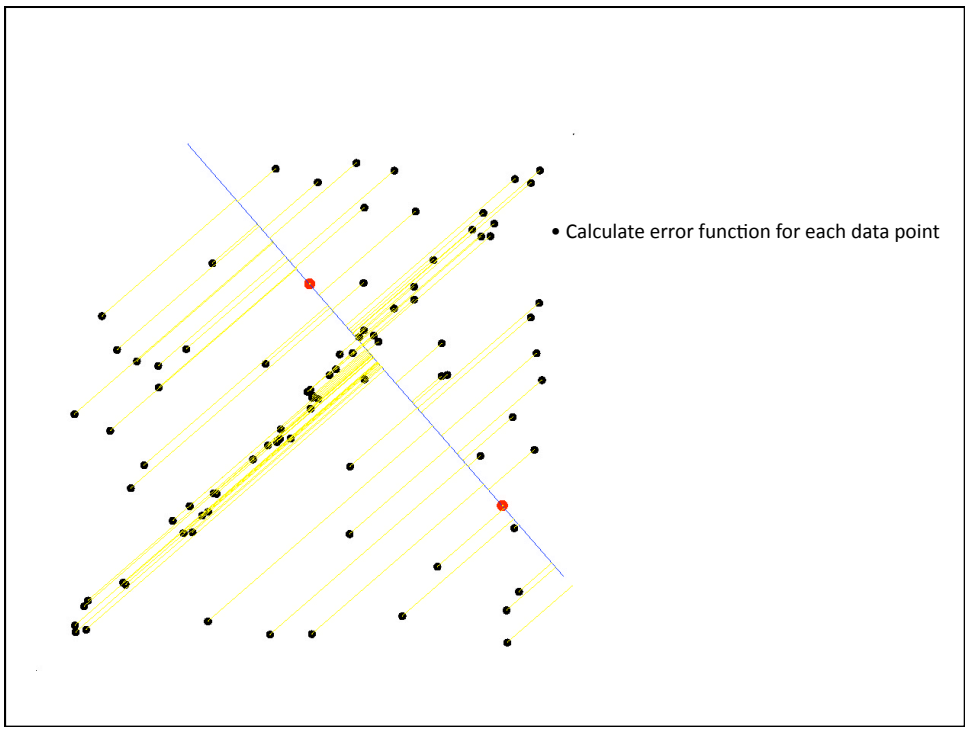
Least squares fit

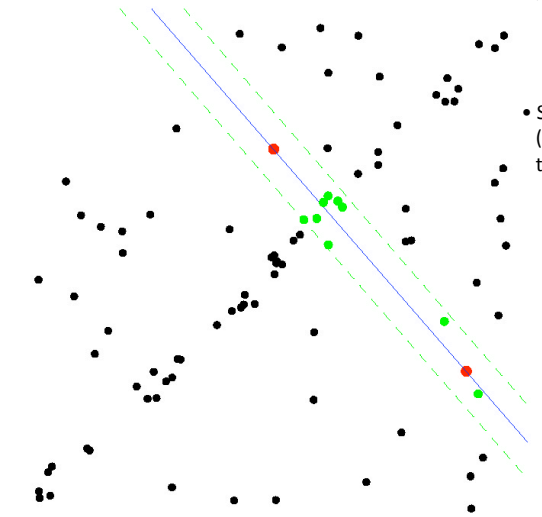
From D. Forsyth



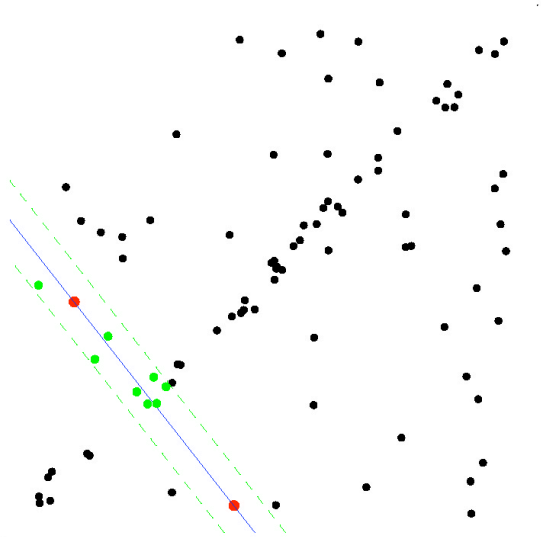
- Select sample of *m* points at random: to estimate a line, 2 points are minimal.


• Calculate model parameters that fit the data in the sample

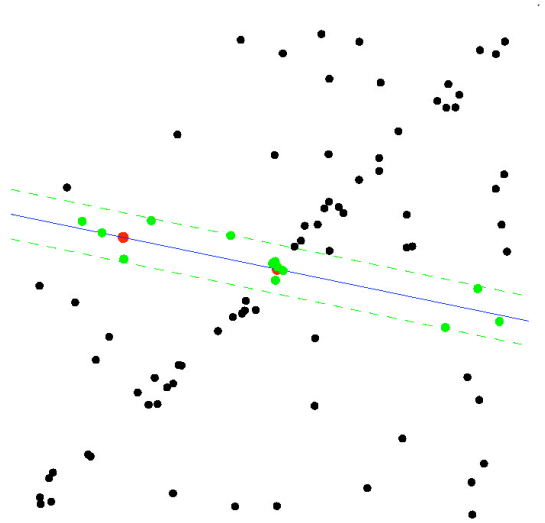

• Calculate error function for each data point



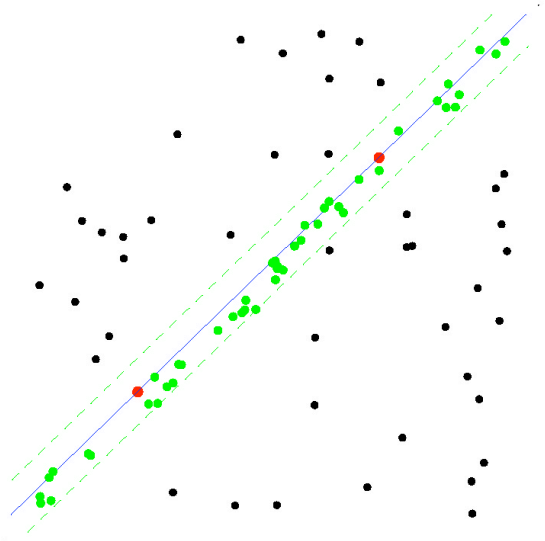
- Select data that support current hypothesis (points with residuals less than some threshold are classified as its inliers)



- Repeat sampling
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis (points with residuals less than some threshold are classified as its inliers)



- Repeat sampling
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis (points with residuals less than some threshold are classified as its inliers)
- .....Repeat until k
- Select the hypothesis with the maximal number of inliers and re-estimate the model parameter using its identified inliers.



Final fitting

Example: fitting affine transformation

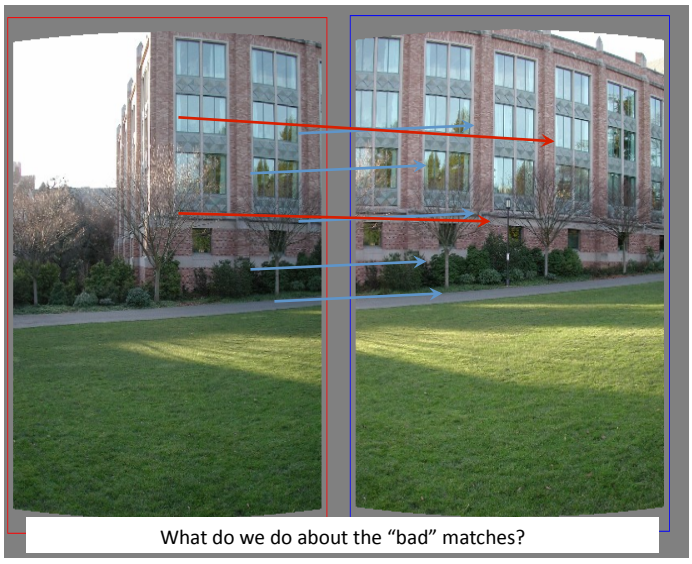
- Affine transform of [x,y] to [u,v]:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- Rewrite to solve for transform parameters (6):

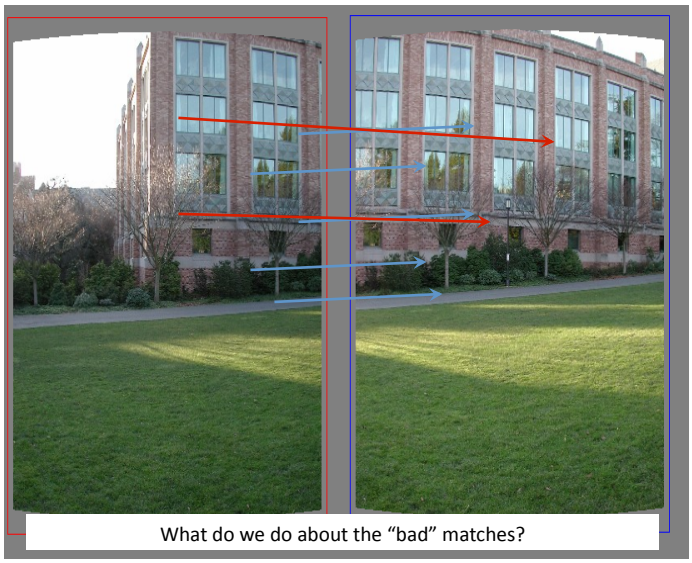
$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

x, y *u, v*



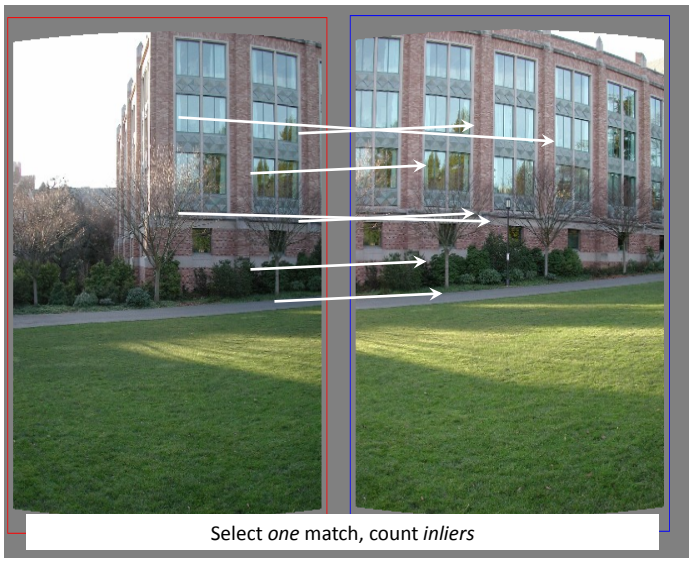
What do we do about the "bad" matches?

From S. Seitz, R. Szeliski and A. Efros

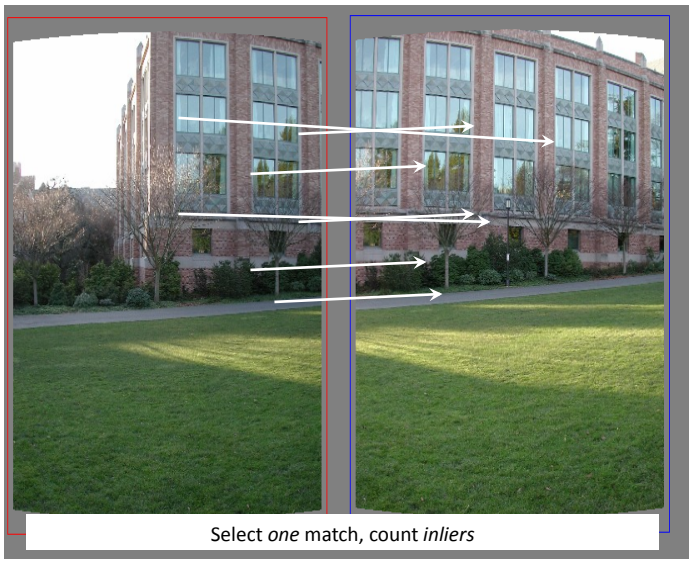


What do we do about the “bad” matches?

From S. Seitz, R. Szeliski and A. Efros

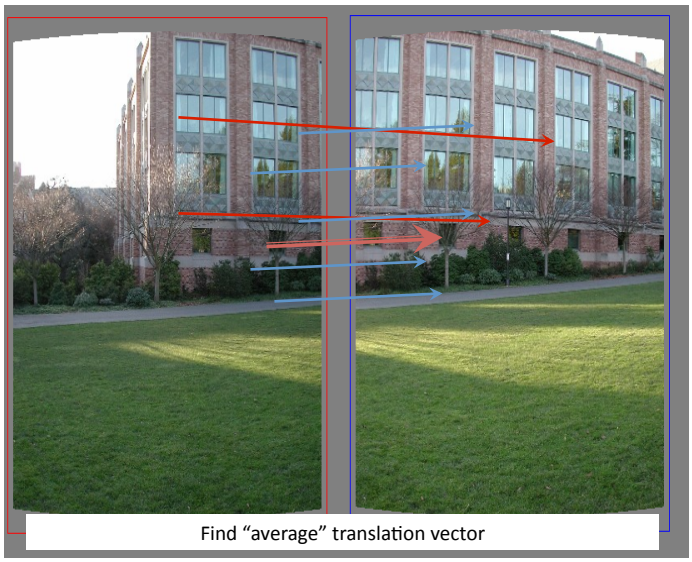


Select *one* match, count *inliers*



Select *one* match, count *inliers*

Least squares fit



Find “average” translation vector

How many iterations / samples?

- Suppose :

w	*number of inliers in data / number of points in data*
n	*is the number of points needed for estimating a model selected independently,*
w^n	*is the probability that all n points are inliers*
$1 - w^n$	*is the probability that at least one of the n points is an outlier*
p	the probability that the algorithm produces a useful resul

- Theoretically the number of iterations *k* (the number of samples) can be obtained from:

$$k = \frac{\log(1-p)}{\log(1-w^n)}$$

It must be chosen high enough to keep this below desired failure rate.

- Theoretical number of samples *k* needed to ensure 95% confidence that at least one outlier free sample:

sample size (# points taken at random)

Proportion of inliers *w* [%]

	15%	20%	30%	40%	50%	70%
2	132	73	32	17	10	4
4	5916	1871	368	116	46	11
7	$1.75 \cdot 10^6$	$2.34 \cdot 10^5$	$1.37 \cdot 10^4$	1827	382	35
8	$1.17 \cdot 10^7$	$1.17 \cdot 10^6$	$4.57 \cdot 10^4$	4570	765	50
12	$2.31 \cdot 10^{10}$	$7.31 \cdot 10^8$	$5.64 \cdot 10^6$	$1.79 \cdot 10^5$	$1.23 \cdot 10^4$	215
18	$2.08 \cdot 10^{15}$	$1.14 \cdot 10^{13}$	$7.73 \cdot 10^9$	$4.36 \cdot 10^7$	$7.85 \cdot 10^5$	1838
30	∞	∞	$1.35 \cdot 10^{16}$	$2.60 \cdot 10^{12}$	$3.22 \cdot 10^9$	$1.33 \cdot 10^5$
40	∞	∞	∞	$2.70 \cdot 10^{16}$	$3.29 \cdot 10^{12}$	$4.71 \cdot 10^6$

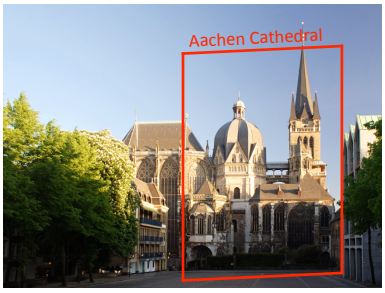
- For data with many outliers, the required number of samples increases dramatically.
In practice the theoretical estimates are optimistic and the actual number of required samples is much higher.

RANSAC summary

- Advantages
  - General method suited to large range of problems
  - Easy to implement
  - Independent of number of dimensions
- Disadvantages
  - Only handles moderate number of outliers (<50%)
  - No upper bound exists on the time required to compute these parameters
  - Requires a large number of samples for data with many outliers thus heavy computation
  - Needs to know the outlier ratio to estimate the number of samples. If no prior information, a conservative number need to be used, for instance, 60% for wide baseline matching results to ensure successful run.
  - Requires a threshold for determining whether points are inliers
- Various improvements to standard approach [Nister, 2004; Matas 2005, Sutter 2005, …….]. Many variants available
  - PROSAC: Progressive RANSAC [Chum, 2005]
  - Preemptive RANSAC [Nister, 2005]
  - ….

Example Applications

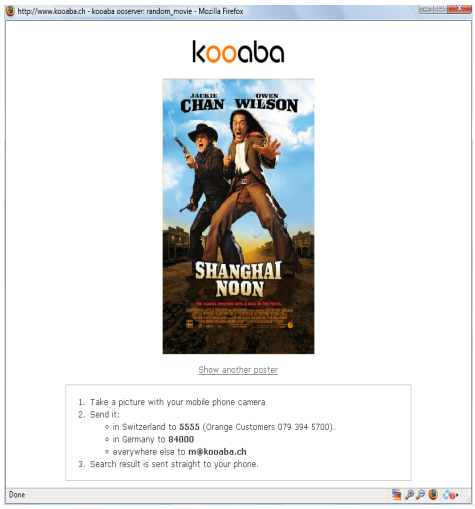
Mobile tourist guide self-localization, object/building recognition, photo/video augmentation



From Quack, Leibe, Van Gool, CIVR'08

Movie Poster Recognition

Content-based retrieval from mobile phone



From Quack, Leibe, Van Gool, CIVR'08

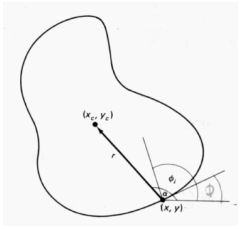
Image Auto-Annotation

Left: Wikipedia image
Right: closest match from Flickr

From Quack, Leibe, Van Gool, CIVR'08

Generalized Hough Transform

- If there is the need to recognize clusters of just a small number of consistent features among a large number of feature match hypotheses (e.g. 1% vs 99%) RANSAC does not work. Generalized Hough transform is the appropriate solution.
- Generalization of Hough transform for an arbitrary contour or shape [Ballard, 1981]
  - Choose reference point for the contour (e.g. the center)
  - For each point on the contour remember where it is located w.r.t. to the reference point (i.e. remember radius r and angle ϕ relative to the contour tangent)
  - Recognition: whenever you find a contour point, calculate the tangent angle and 'vote' for all possible reference points



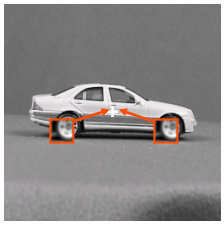
- To assess a transformation between images, the same idea can apply to local features versus a transformation

The Generalized Hough algorithm

- The key to efficiency is to have each feature determine as many parameters as possible
  - For example, lines can be detected much more efficiently from small edge elements (or points with local gradients) than from just points
  - For object recognition, each feature should predict location, scale, and orientation
- Examine all clusters in Hough transform with at least m features
- Perform least-squares affine fit to model.
- Discard outliers and perform top-down check for additional features.
- Evaluate probability that match is correct
  - Use Bayesian model, with probability that features would arise by chance if object was *not* present
  - Takes account of object size in image, textured regions, model feature count in database, accuracy of fit (D. Lowe, 2001)
- The Hough transform can extract feature groupings from clutter in linear time

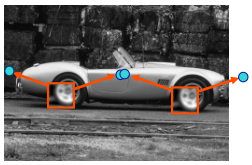
Example: recognition with local features

- For every feature, store all possible "occurrences"



- Object identity
- Pose
- Relative position

For new image, let the matched features vote for possible object positions



From K. Grauman, B. Leibe

Example Applications

Large-Scale Retrieval

Query

Results from 5k Flickr images (demo available for 100k set)

From Philbin CVPR'07

Planar recognition

- Models for planar surfaces with SIFT keys
- Planar surfaces can be reliably recognized at a rotation of 60° away from the camera
- Affine fit approximates perspective projection
- Only 3 points are needed for recognition

From K. Grauman, B. Leibe

3D Object Recognition

- Extract outlines with background subtraction
- Only 3 keypoints are needed for recognition, so extra keypoints provide robustness
- Affine model is no longer as accurate

From K. Grauman, B. Leibe