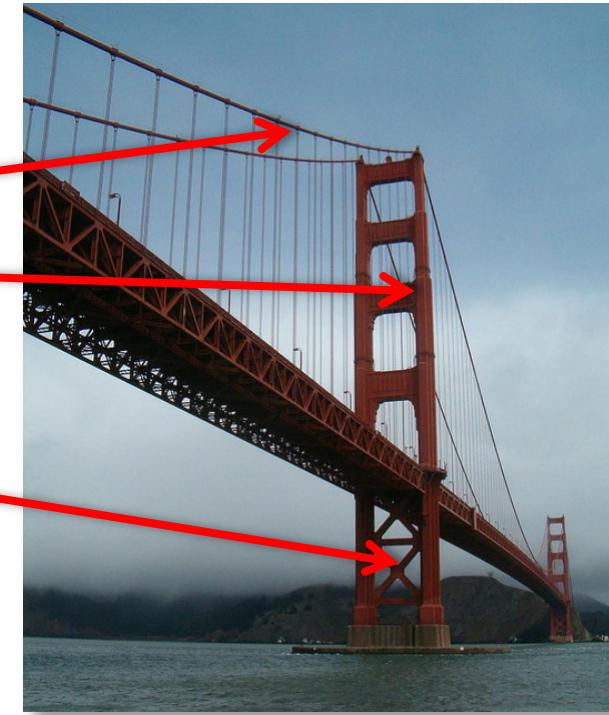
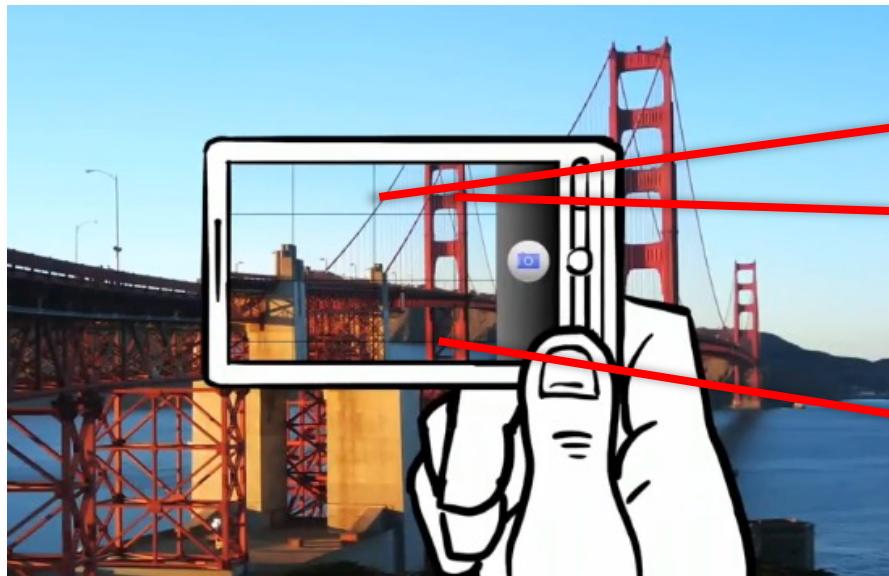


Feature descriptors and matching

Slide Credit: Richard Szeliski



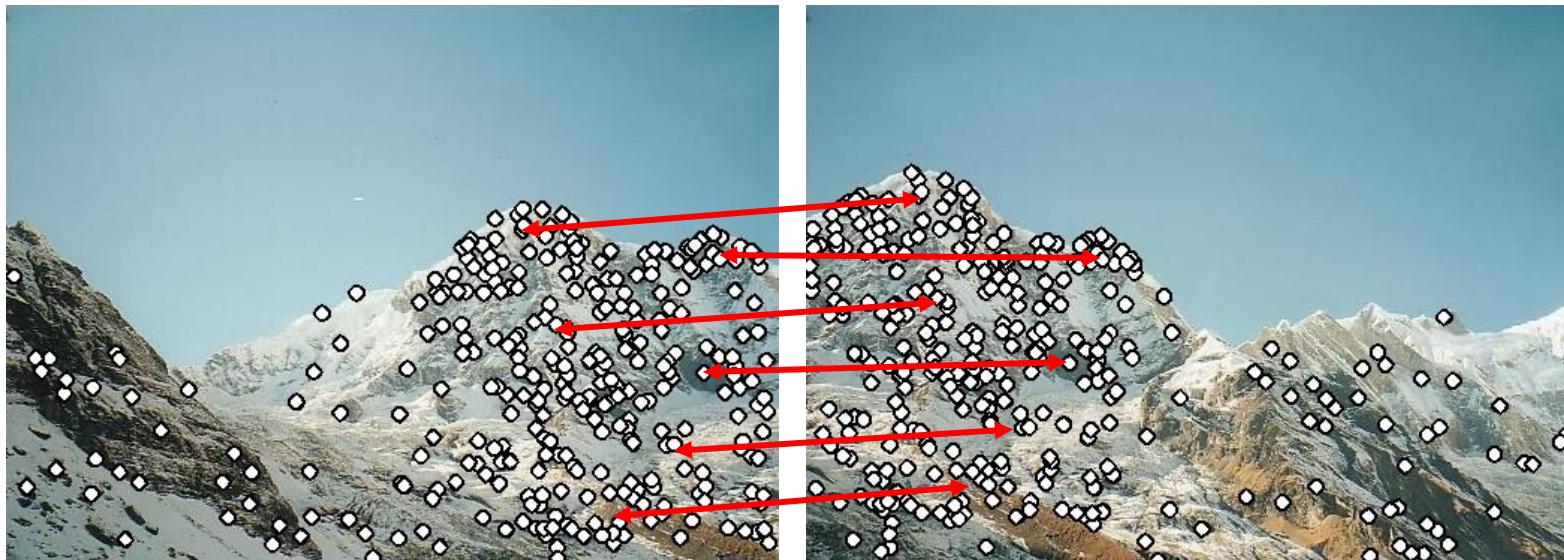
Why extract features?

- Motivation: panorama stitching
 - We have two images – how do we combine them?



Why extract features?

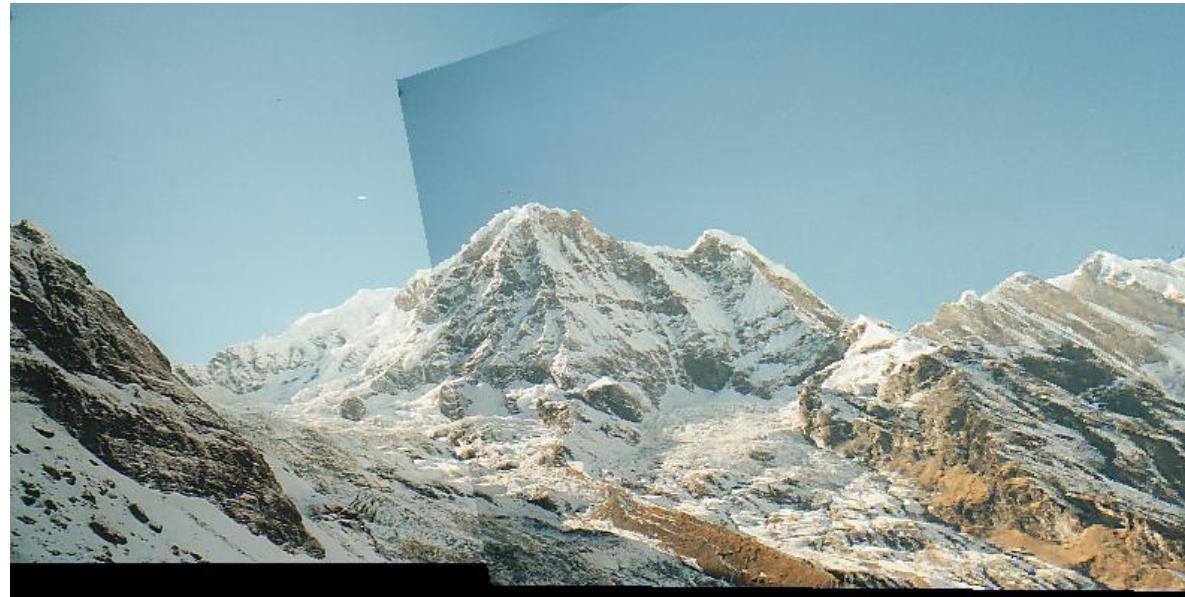
- Motivation: panorama stitching
 - We have two images – how do we combine them?



Step 1: extract features
Step 2: match features

Why extract features?

- Motivation: panorama stitching
 - We have two images – how do we combine them?



Step 1: extract features
Step 2: match features
Step 3: align images

Image matching



by [Diva Sian](#)



by [swashford](#)

Harder case

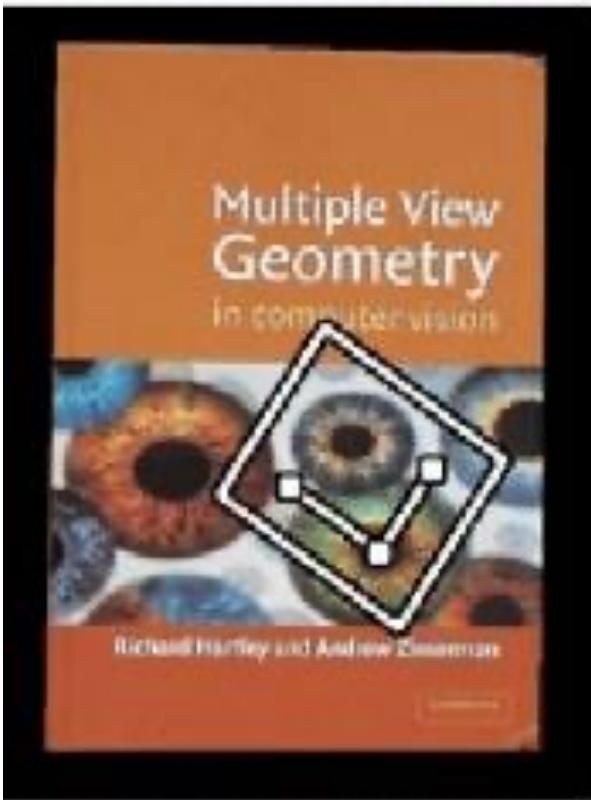


by [Diva Sian](#)



by [scgbt](#)

Feature Matching



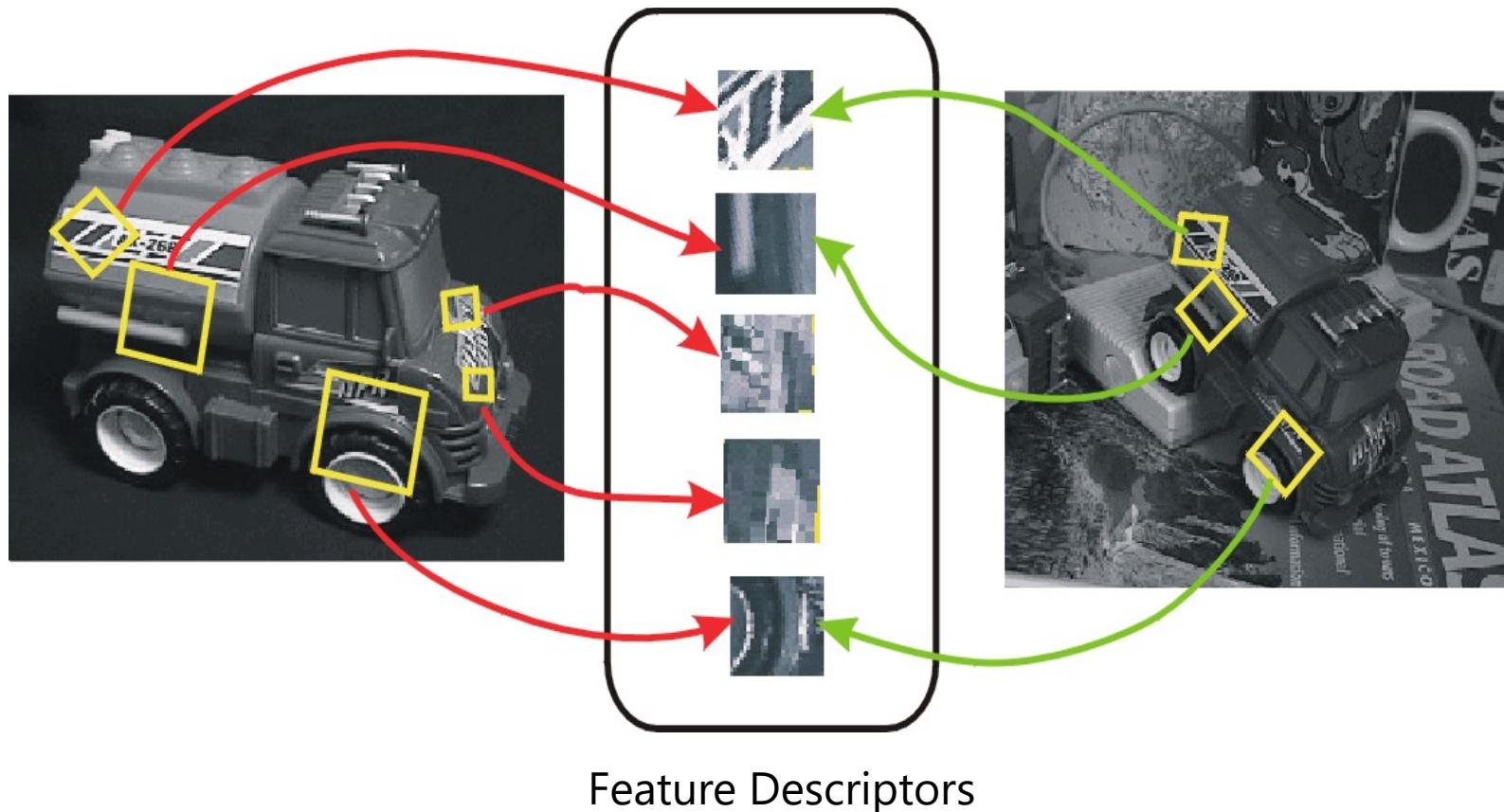
Feature Matching



Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, ...



Advantages of local features

Locality

- features are local, so robust to occlusion and clutter

Quantity

- hundreds or thousands in a single image

Distinctiveness:

- can differentiate a large database of objects

Efficiency

- real-time performance achievable

More motivation...

Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking (e.g. for AR)
- Object recognition
- Image retrieval
- Robot/car navigation
- ... other

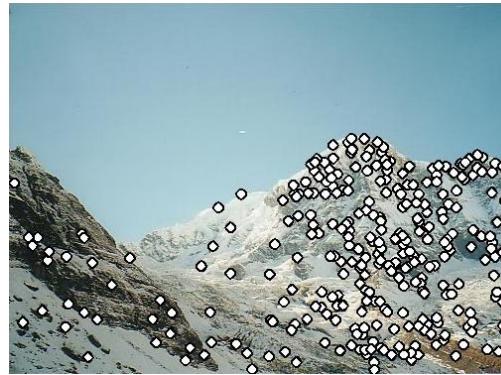
Approach

- 1. Feature detection:** find it
- 2. Feature descriptor:** represent it
- 3. Feature matching:** match it

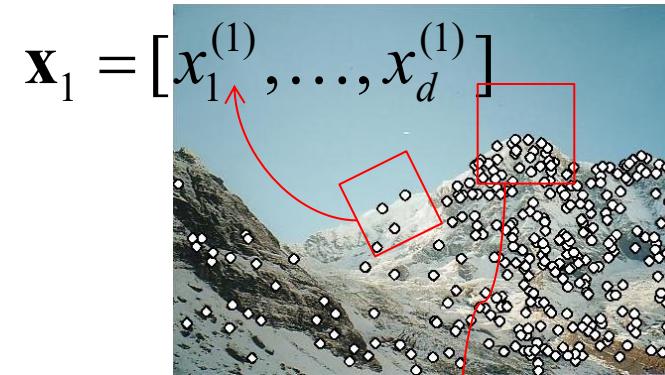
Feature tracking: track it, when motion

Local features: main components

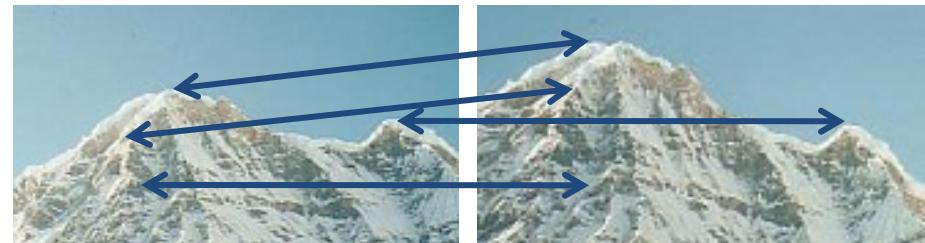
1) Detection: Identify the interest points



2) Description: Extract vector feature descriptor surrounding each interest point.

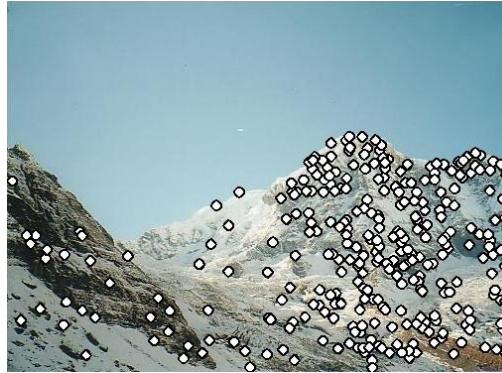


3) Matching: Determine correspondence between descriptors in two views

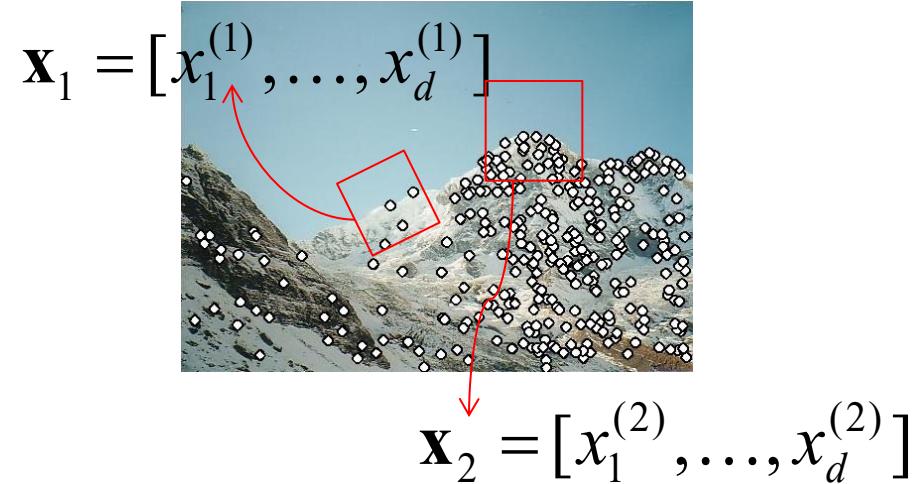


Local features: main components

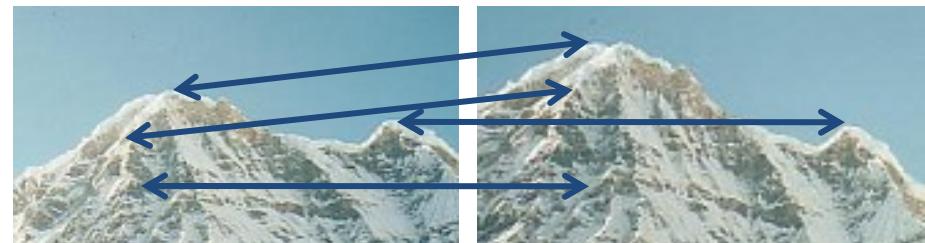
- 1) Detection: Identify the interest points



- 2) **Description:** Extract vector feature descriptor surrounding each interest point.



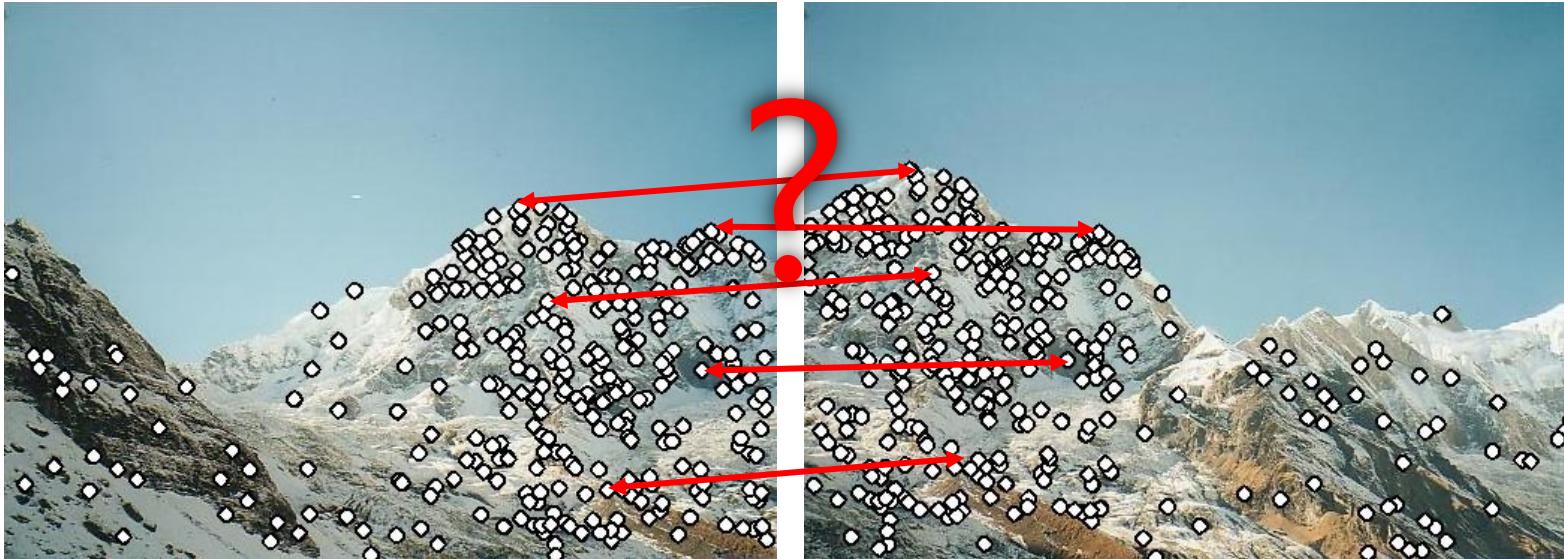
- 3) Matching: Determine correspondence between descriptors in two views



Feature descriptors

We know how to detect good points

Next question: **How to match them?**

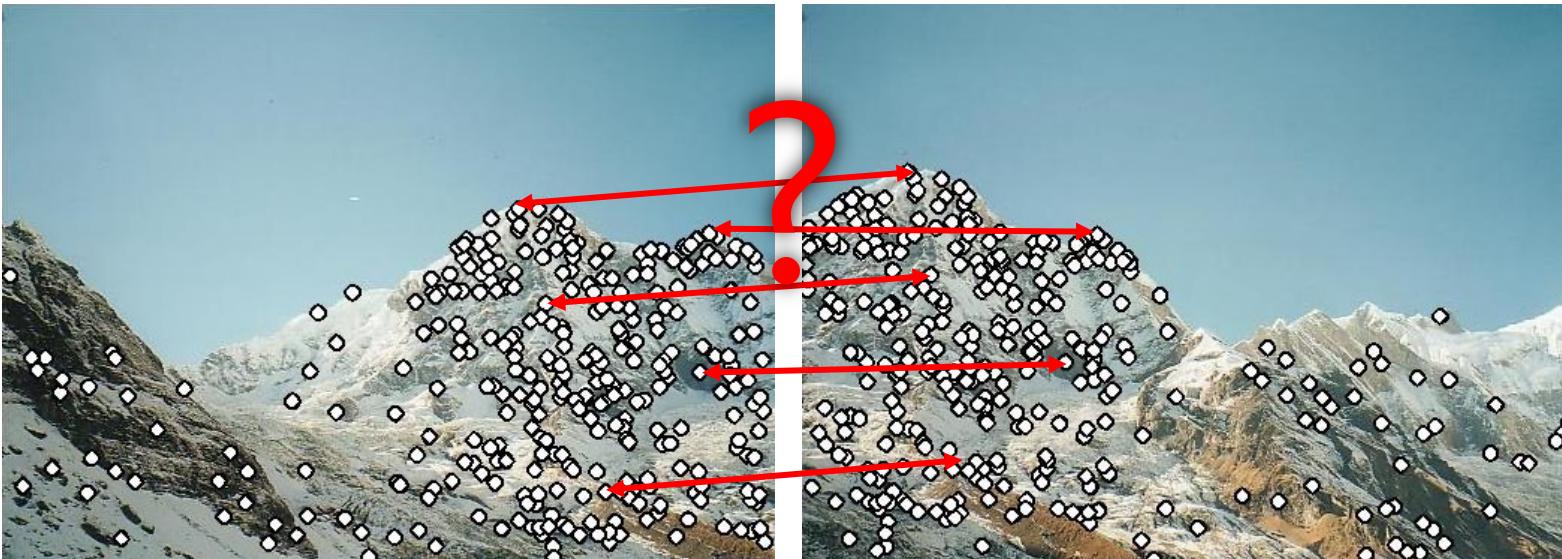


Answer: Come up with a *descriptor* for each point, find similar descriptors between the two images

Feature descriptors

We know how to detect good points

Next question: **How to match them?**



Lots of possibilities

- Simple option: match square windows around the point
- State of the art approach: SIFT
 - David Lowe, UBC <http://www.cs.ubc.ca/~lowe/keypoints/>

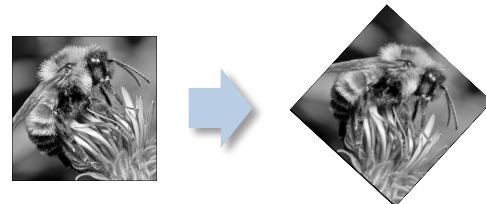
Invariance vs. discriminability

- Invariance:
 - Descriptor shouldn't change even if image is transformed
- Discriminability:
 - Descriptor should be highly unique for each point

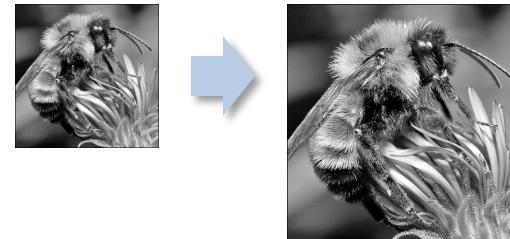
Image transformations revisited

- Geometric

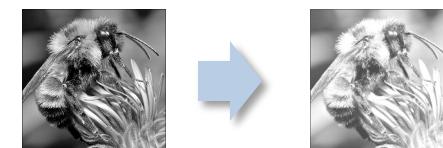
Rotation



Scale



- Photometric
Intensity change



Invariant descriptors

- We looked at invariant / equivariant **detectors**
- Most feature descriptors are also designed to be invariant to:
 - Translation, 2D rotation, scale
- They can usually also handle
 - Limited 3D rotations (SIFT works up to about 60 degrees)
 - Limited affine transforms (some are fully affine invariant)
 - Limited illumination/contrast changes

Rotation invariance for feature descriptors

- Find dominant orientation of the image patch
 - E.g., given by \mathbf{x}_{\max} , the eigenvector of \mathbf{H} corresponding to λ_{\max} (the *larger* eigenvalue)
 - Or simply the orientation of the (smoothed) gradient
 - Rotate the patch according to this angle

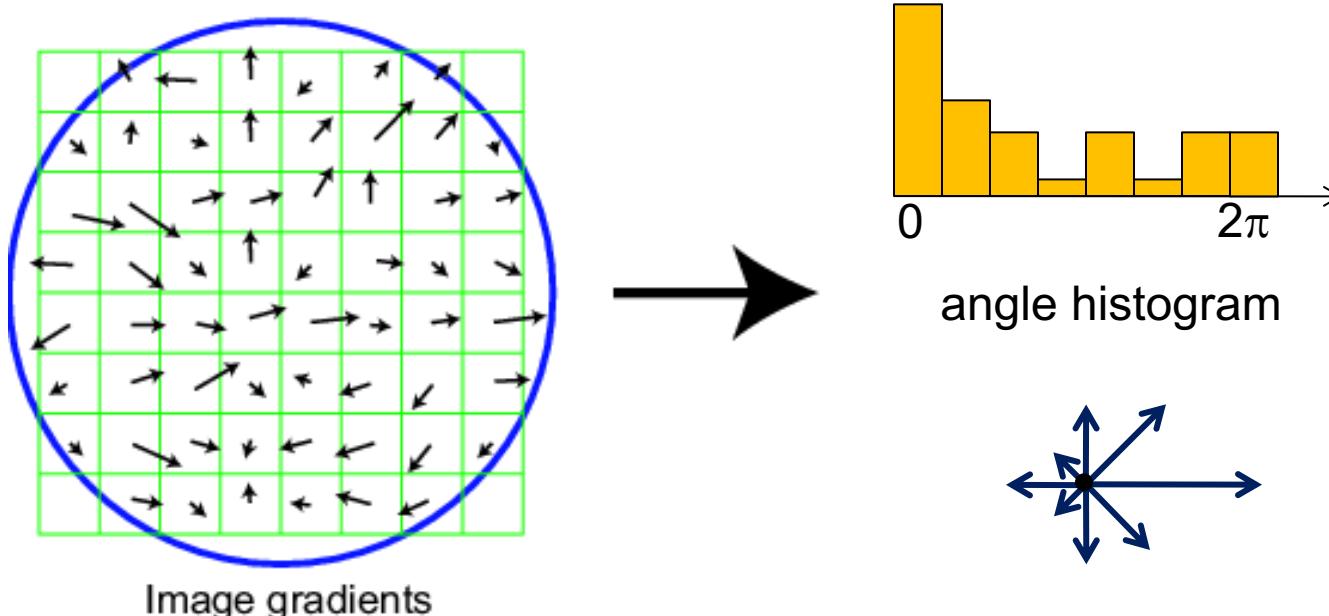


Figure by Matthew Brown

Scale Invariant Feature Transform

Basic idea:

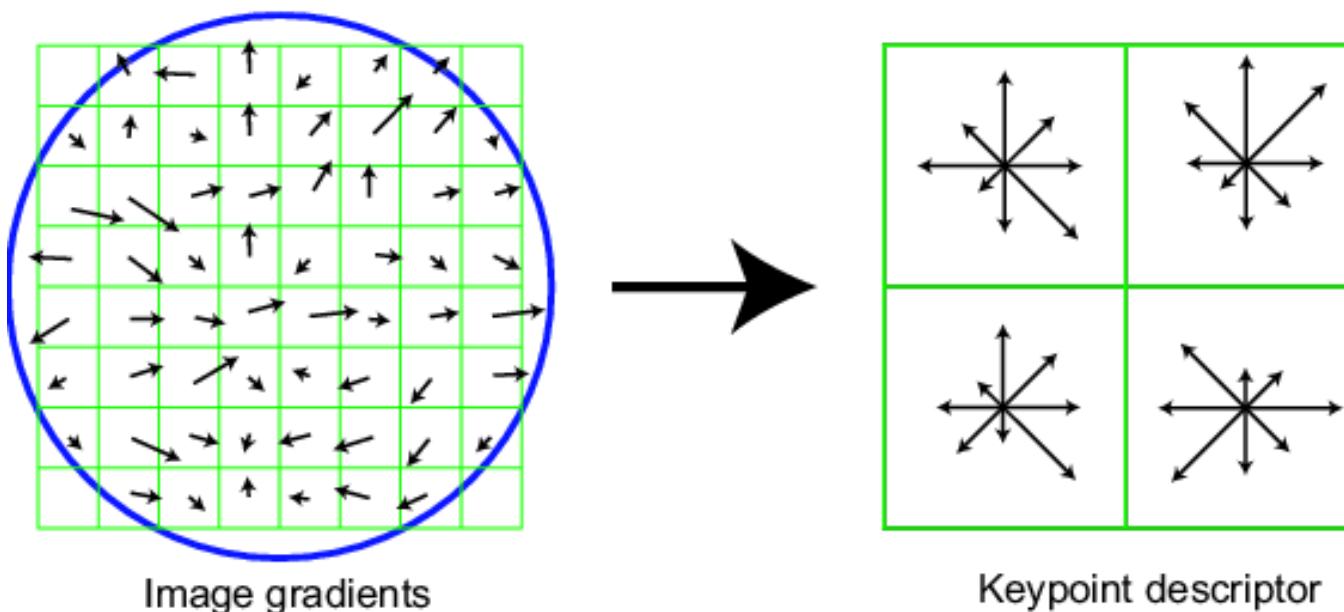
- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations



SIFT descriptor

Full version

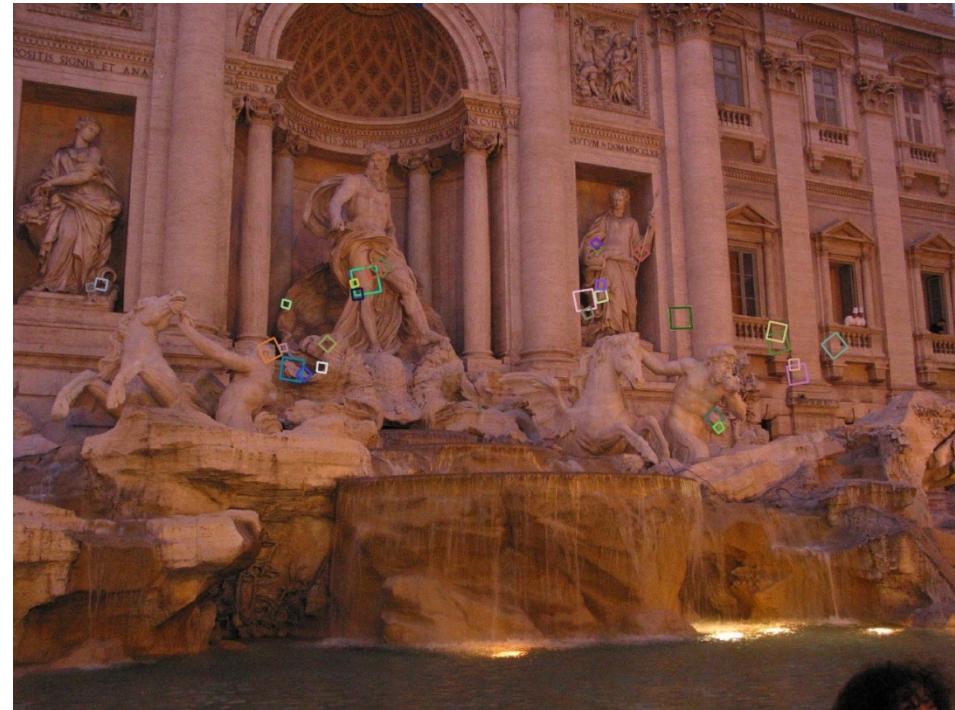
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Properties of SIFT

Extraordinarily robust matching technique

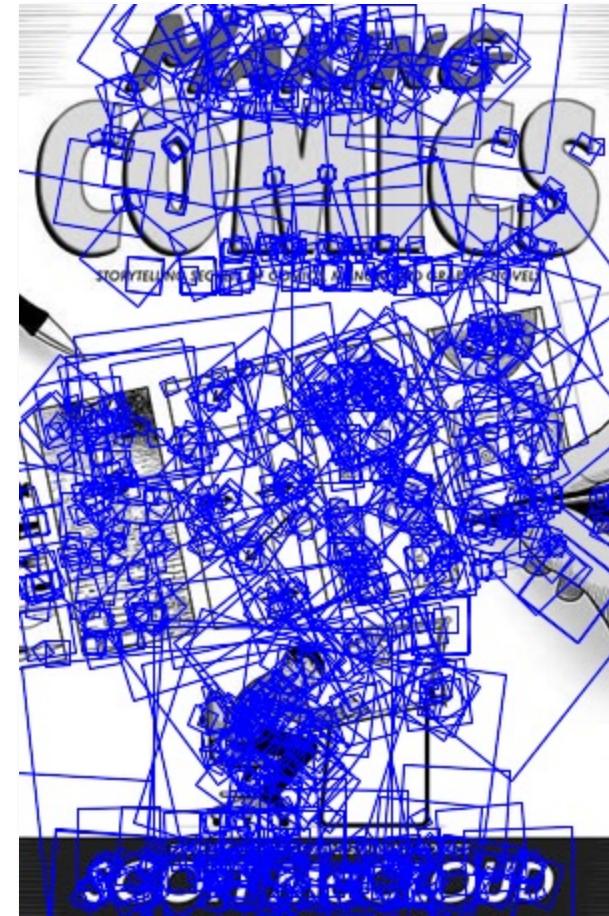
- Can handle changes in viewpoint (up to about 60 degree out of plane rotation)
- Can handle significant changes in illumination (sometimes even day vs. night (below))
- Relatively fast—hard to make real-time, but can run in <1s for moderate image sizes
 - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT



SIFT Example



sift



868 SIFT features

Other descriptors

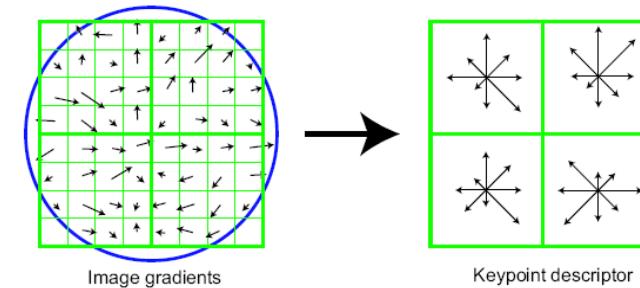
- HOG: Histogram of Gradients (HOG)
 - Dalal/Triggs
 - Sliding window, pedestrian detection
- FREAK: Fast Retina Keypoint
 - Perceptually motivated
 - Used in Visual SLAM
- LIFT: Learned Invariant Feature Transform
 - Learned via deep learning



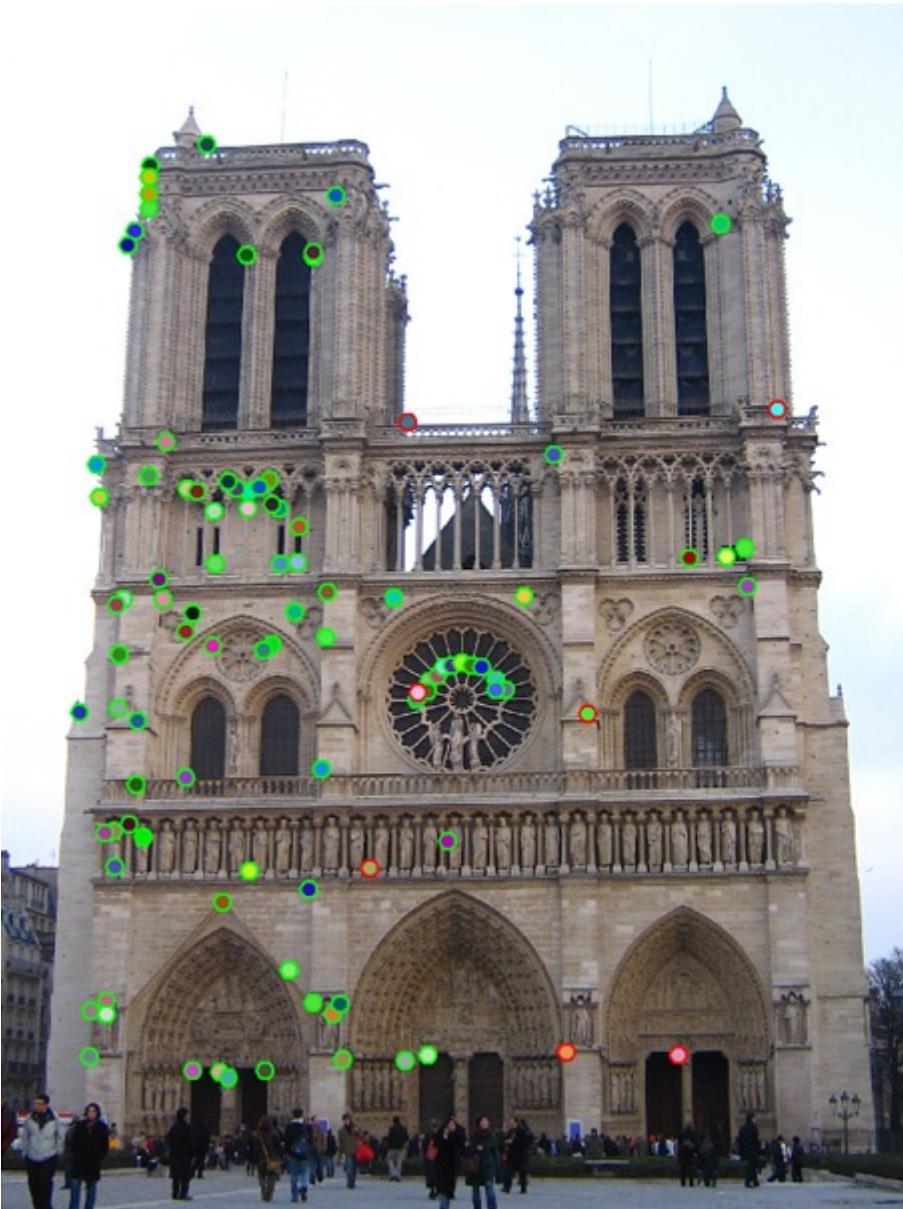
<https://arxiv.org/abs/1603.09114>

Summary

- Keypoint detection: repeatable and distinctive
 - Corners, blobs, stable regions
 - Harris, DoG
- Descriptors: robust and selective
 - spatial histograms of orientation
 - SIFT and variants are typically good for stitching and recognition
 - But, need not stick to one



Which features match?



Feature matching

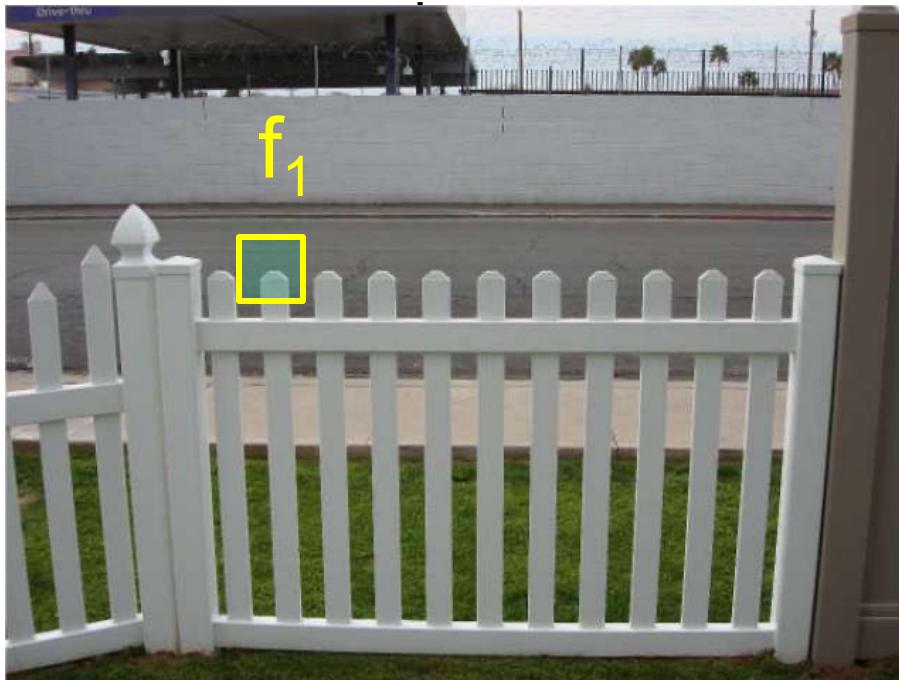
Given a feature in I_1 , how to find the best match in I_2 ?

1. Define distance function that compares two descriptors
2. Test all the features in I_2 , find the one with min distance

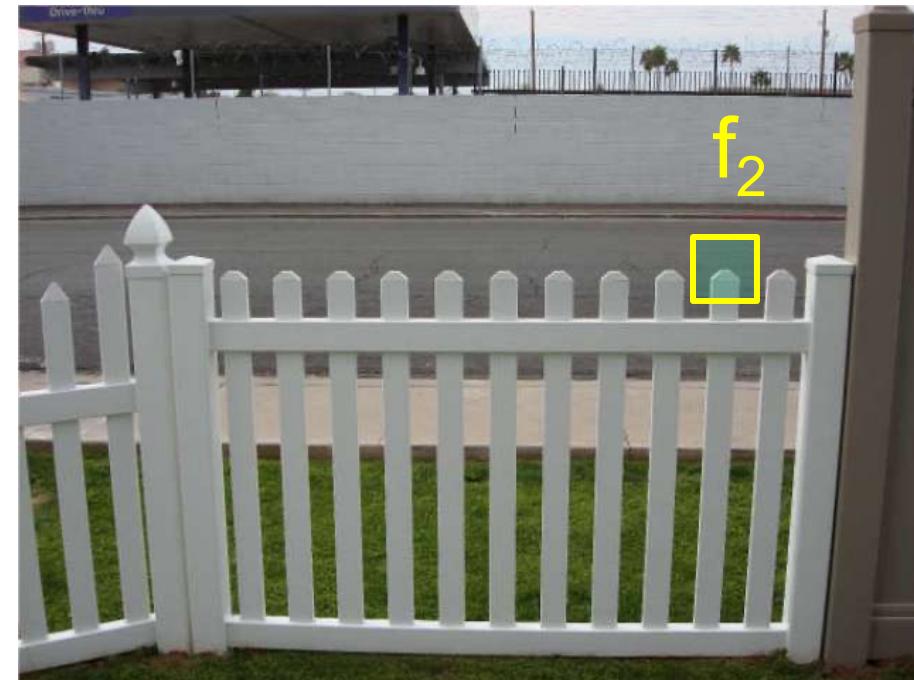
Feature distance

How to define the difference between two features f_1 , f_2 ?

- Simple approach: L_2 distance, $\|f_1 - f_2\|$
- can give small distances for ambiguous (incorrect)



I_1

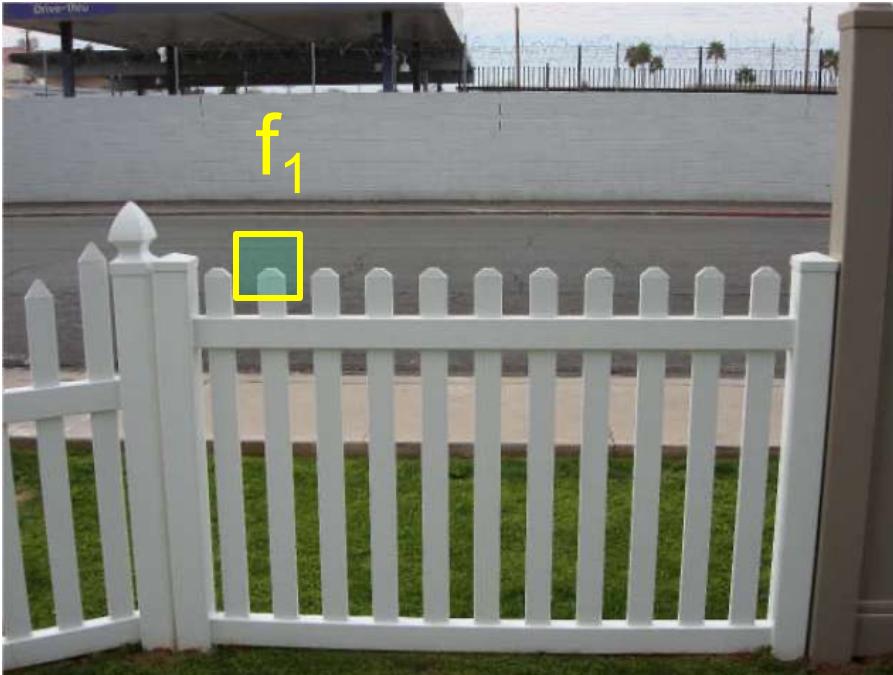


I_2

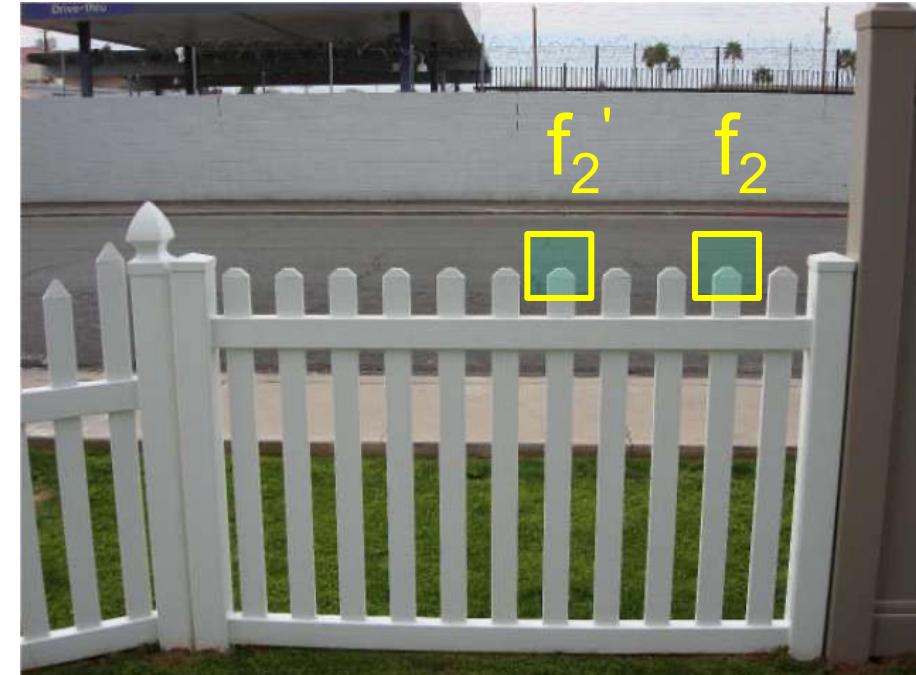
Feature distance

How to define the difference between two features f_1, f_2 ?

- Better approach: ratio distance = $\|f_1 - f_2\| / \|f_1 - f_2'\|$
 - f_2 is best SSD match to f_1 in I_2
 - f_2' is 2nd best SSD match to f_1 in I_2
 - gives large values for ambiguous matches

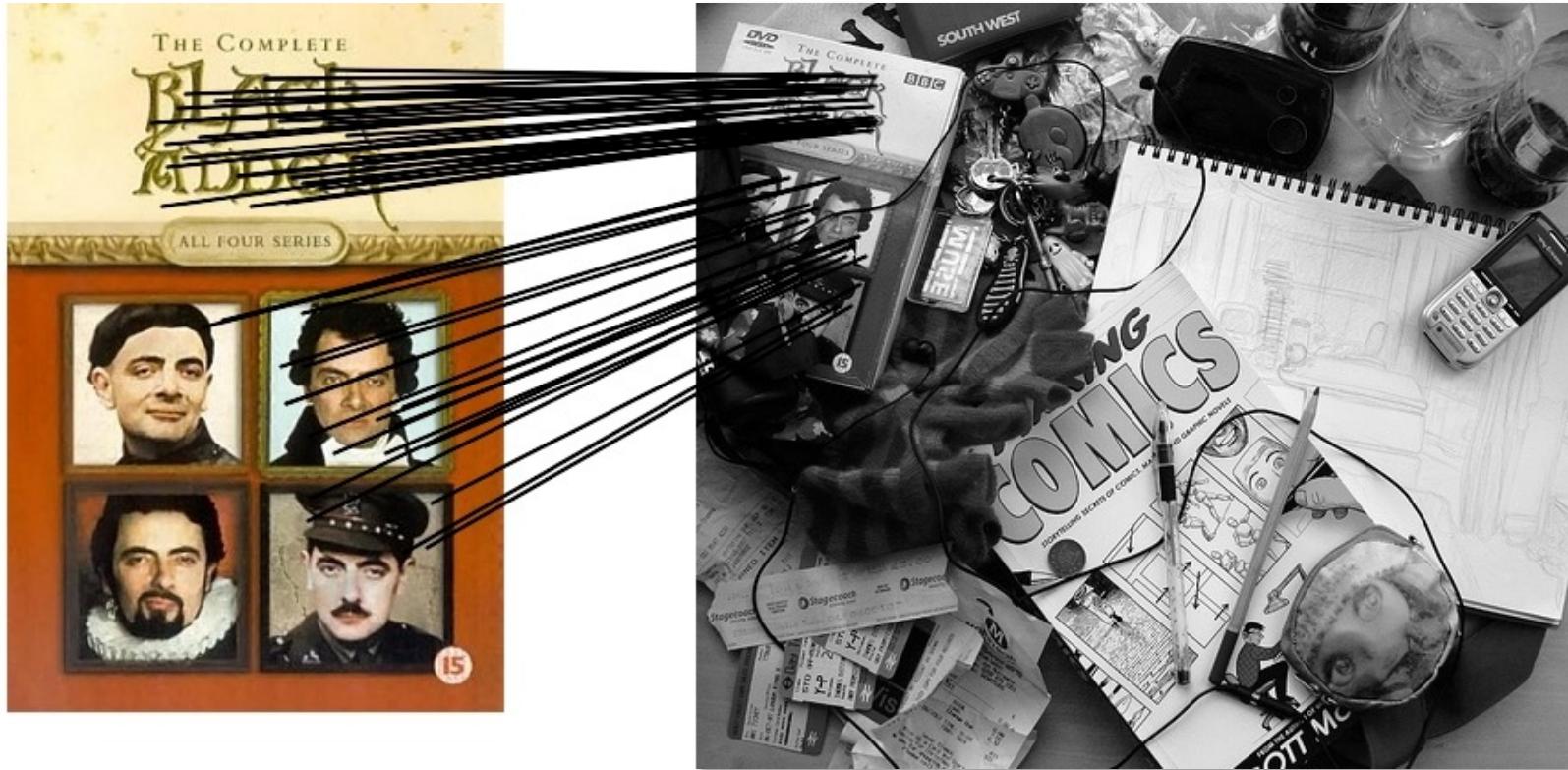


I_1



I_2

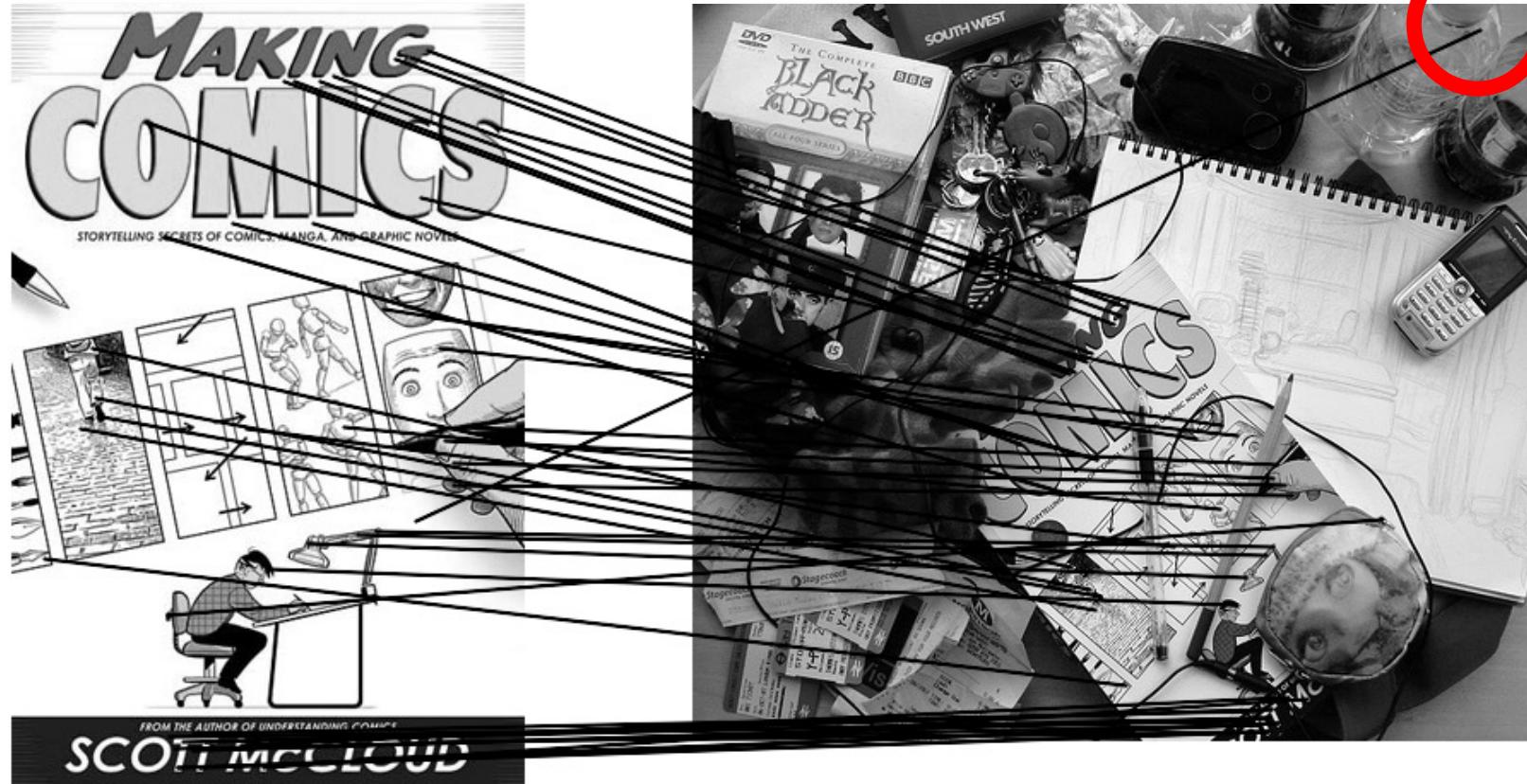
Feature matching example



58 matches (thresholded by ratio score)

Feature matching example

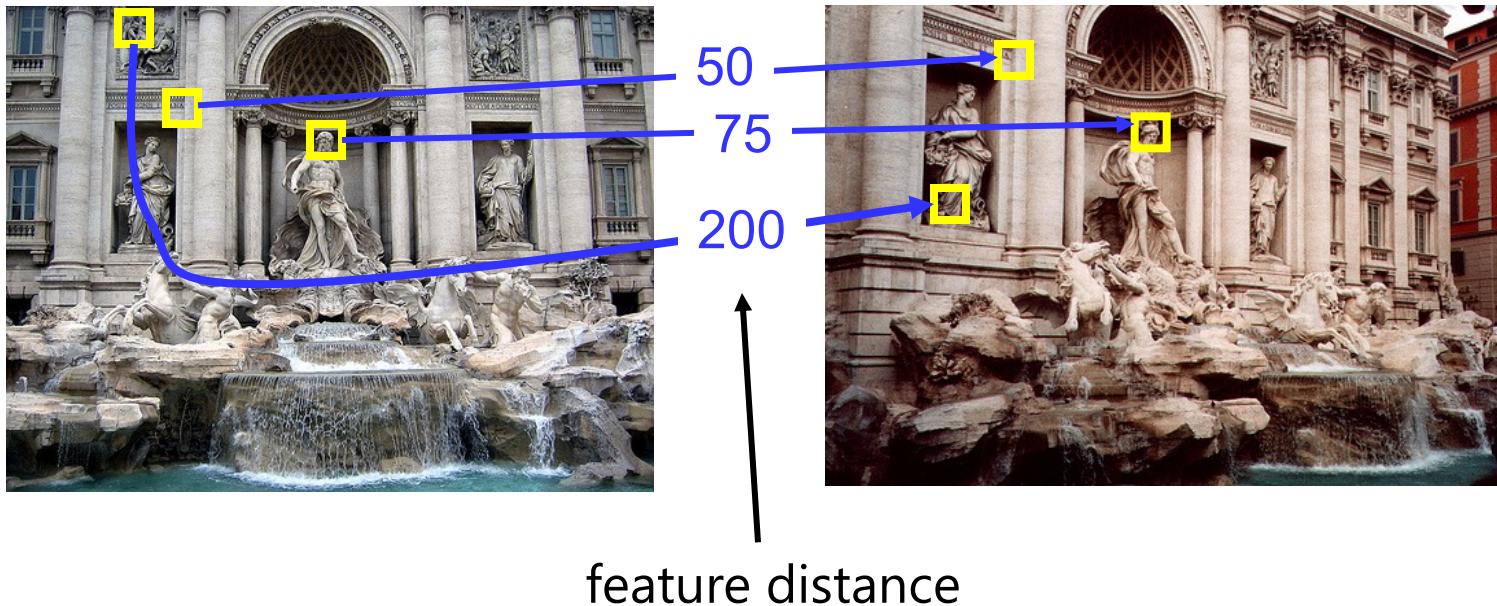
We'll deal
with **outliers**
later



51 matches (thresholded by ratio score)

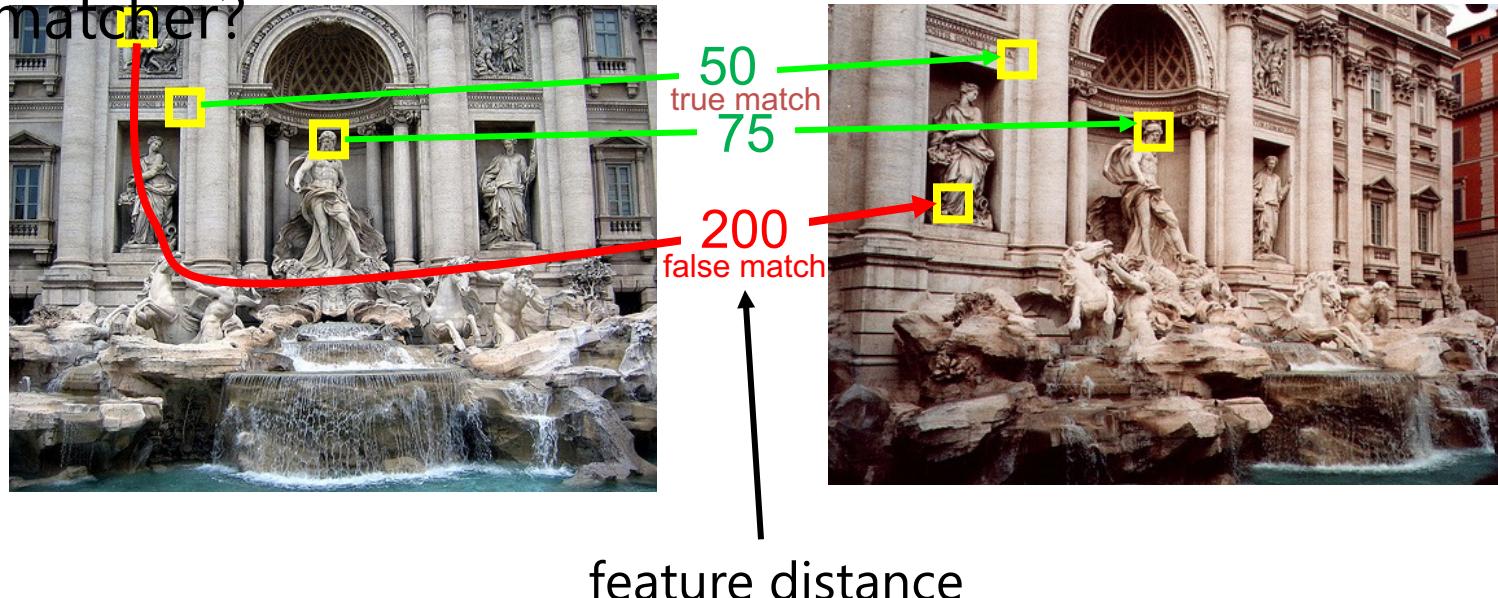
Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

How can we measure the performance of a feature matcher?

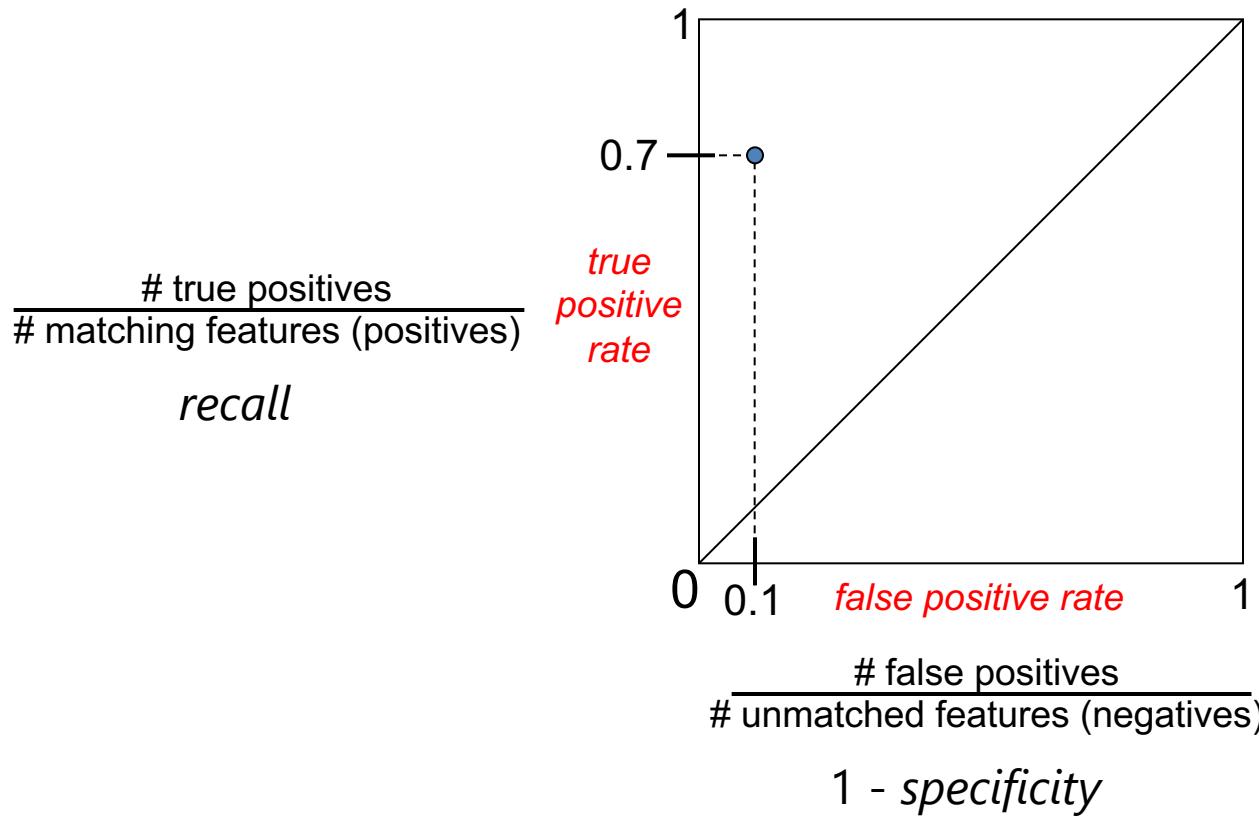


The distance threshold affects performance

- True positives = # of detected matches that are correct
 - Suppose we want to maximize these—how to choose threshold?
- False positives = # of detected matches that are incorrect
 - Suppose we want to minimize these—how to choose threshold?

Evaluating the results

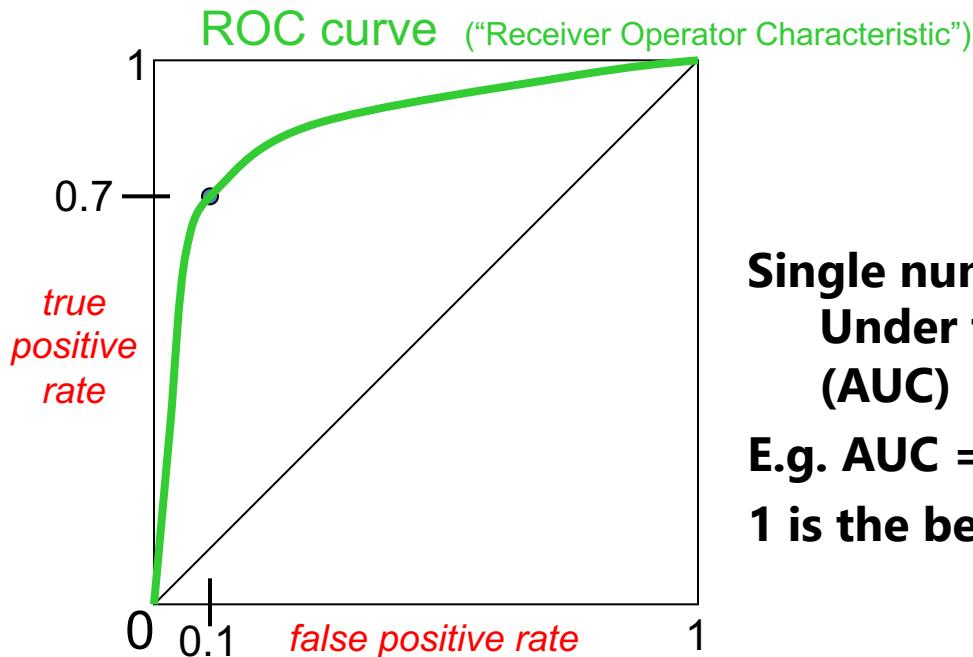
How can we measure the performance of a feature matcher?



Evaluating the results

How can we measure the performance of a feature matcher?

$$\frac{\# \text{ true positives}}{\# \text{ matching features (positives)}} \\ \textit{recall}$$



$$\frac{\# \text{ false positives}}{\# \text{ unmatched features (negatives)}} \\ 1 - \textit{specificity}$$

Single number: Area Under the Curve (AUC)
E.g. AUC = 0.87
1 is the best

More on feature detection/description

<http://www.robots.ox.ac.uk/~vgg/research/affine/>

<http://www.cs.ubc.ca/~lowe/keypoints/>

<http://www.vision.ee.ethz.ch/~surf/>

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: [K. Mikolajczyk](#) and [C. Schmid](#), Scale and Affine invariant interest point detectors. In IJC V 60(1):63-86, 2004. [PDF](#)
- *MSER*: [J.Matas](#), [O. Chum](#), [M. Urban](#), and [T. Pajdla](#), Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. [PDF](#)
- *IBR & EBR*: [T.Tuytelaars](#) and [L. Van Gool](#), Matching widely separated views based on affine invariant regions . In IJCV 59(1):61-85, 2004. [PDF](#)
- *Salient regions*: [T. Kadir](#), [A. Zisserman](#), and [M. Brady](#), An affine invariant salient region detector. In ECCV p. 404-416, 2004. [PDF](#)
- *All Detectors - Survey*: [T. Tuytelaars](#) and [K. Mikolajczyk](#), Local Invariant Feature Detectors - Survey. In CVG, 3(1):1-110, 2008. [PDF](#)

Region descriptors

- *SIFT*: [D. Lowe](#), Distinctive image features from scale invariant keypoints. In IJCV 60(2):91-110, 2004. [PDF](#)

Performance evaluation

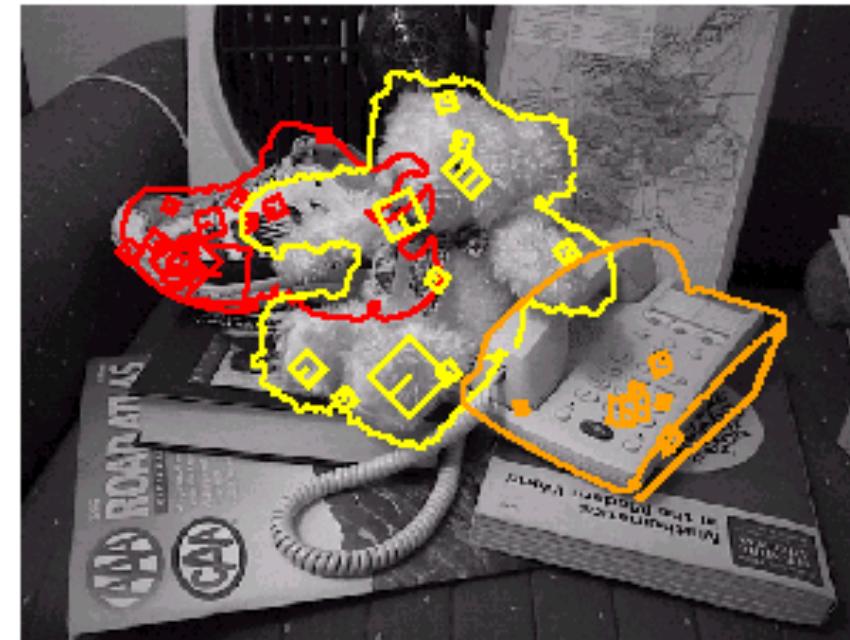
- [K. Mikolajczyk](#), [T. Tuytelaars](#), [C. Schmid](#), [A. Zisserman](#), [J. Matas](#), [F. Schaffalitzky](#), [T. Kadir](#) and [L. Van Gool](#), A comparison of affine region detectors. In IJCV 65(1/2):43-72, 2005. [PDF](#)
- [K. Mikolajczyk](#), [C. Schmid](#), A performance evaluation of local descriptors. In PAMI 27(10):1615-1630 . [PDF](#)

Lots of applications

Features are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- ... other

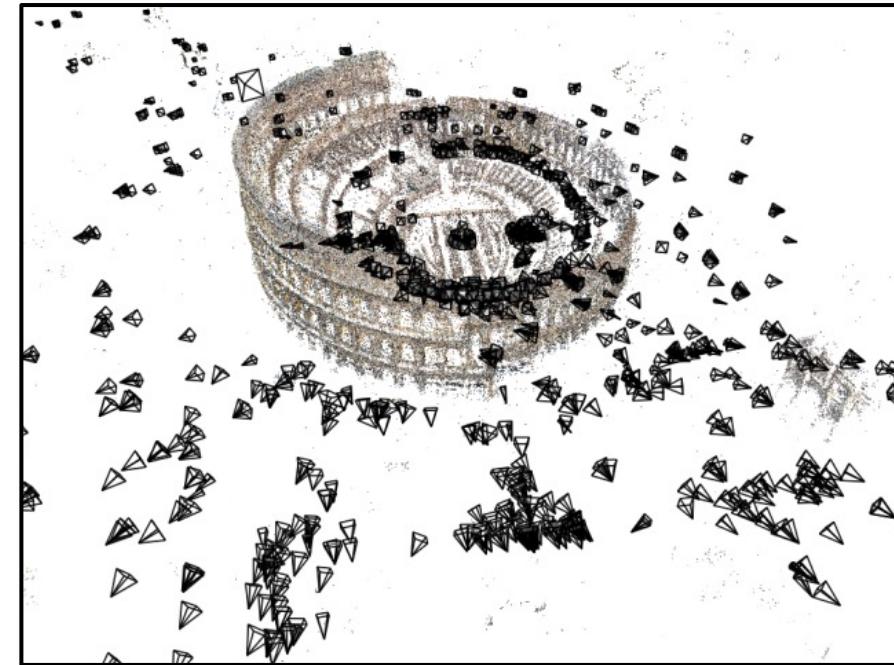
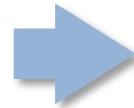
Object recognition (David Lowe)



3D Reconstruction



Internet Photos ("Colosseum")



Reconstructed 3D cameras and points

Augmented Reality

