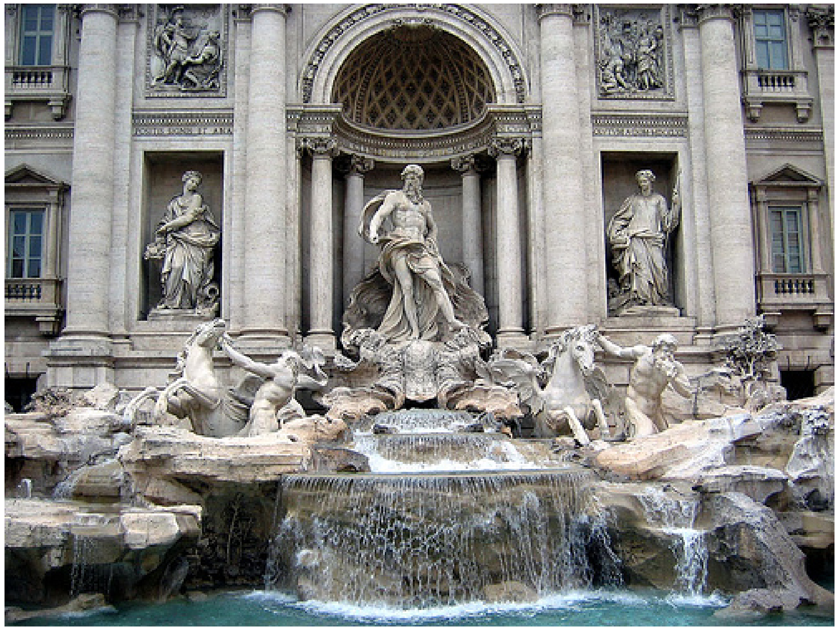
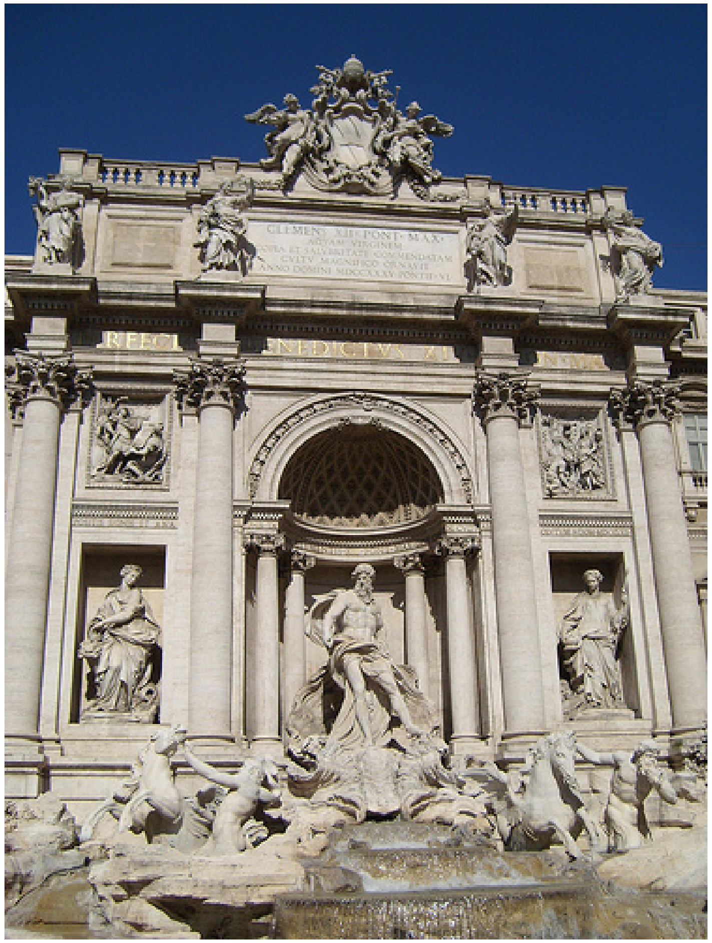
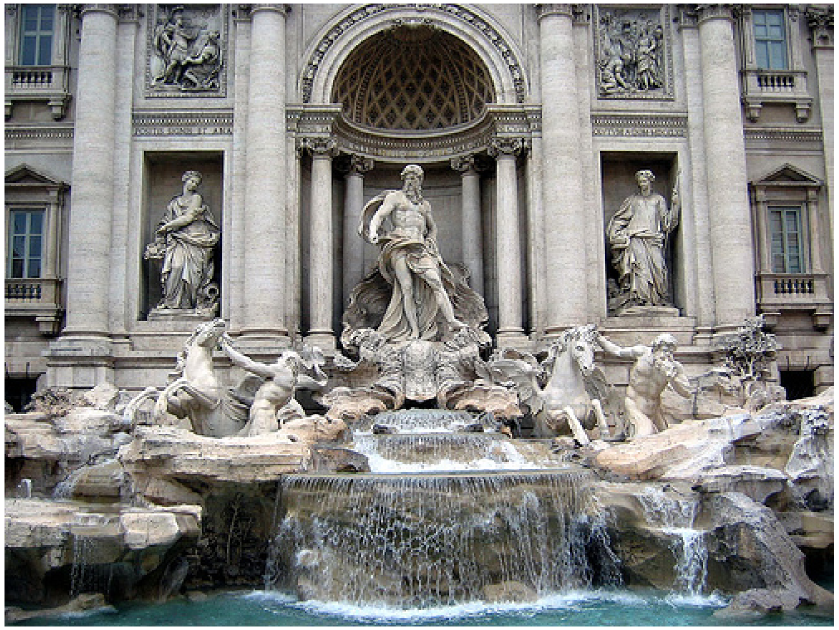
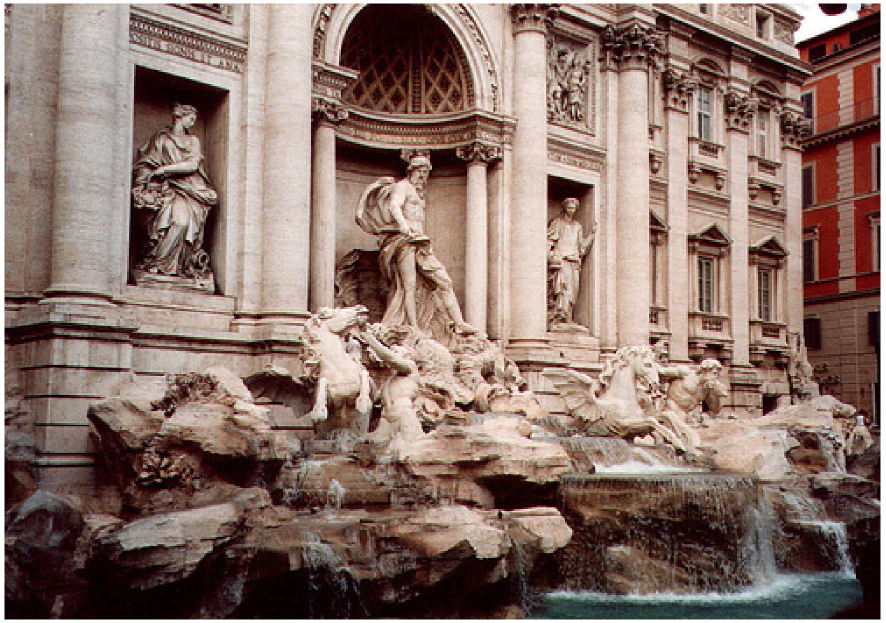
Image matching


by Diva Sian


by swashford

Harder case

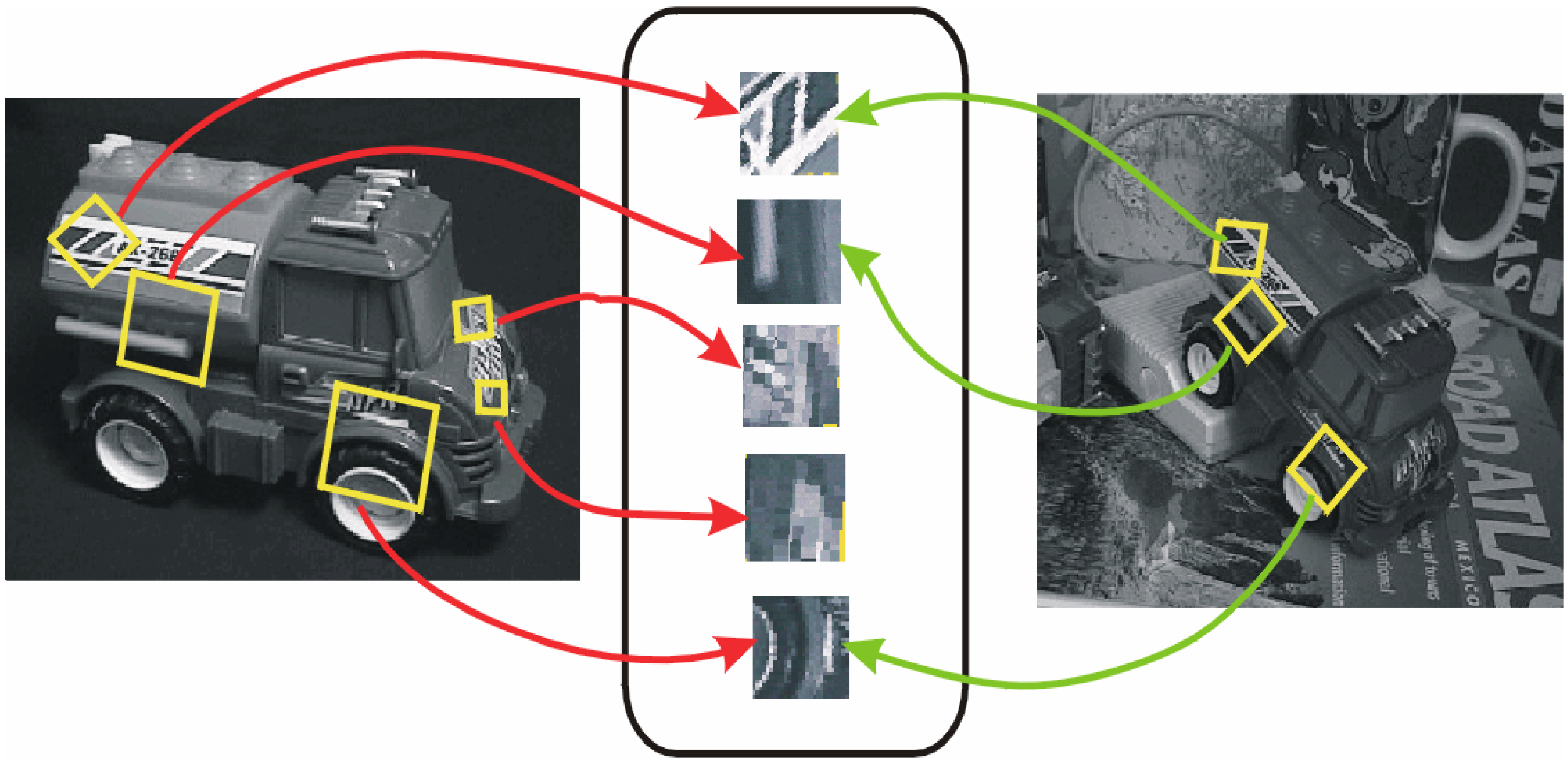

by Diva Sian


by scgbt

Invariant local features

Find features that are invariant to transformations

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

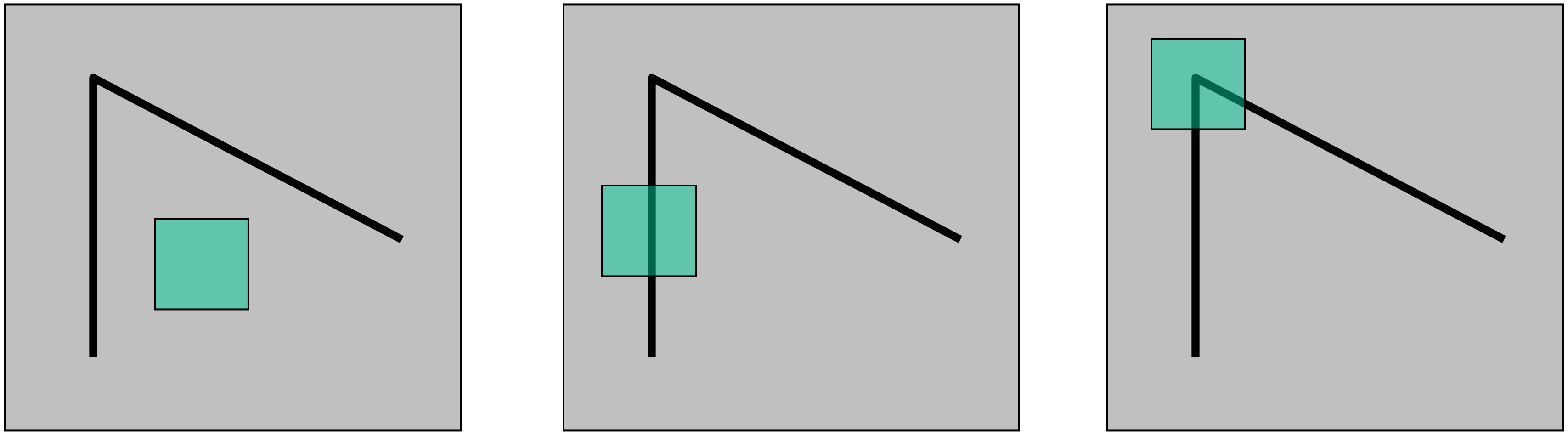


Feature Descriptors

Local measures of uniqueness

Suppose we only consider a small window of pixels

- What defines whether a feature is a good or bad candidate?

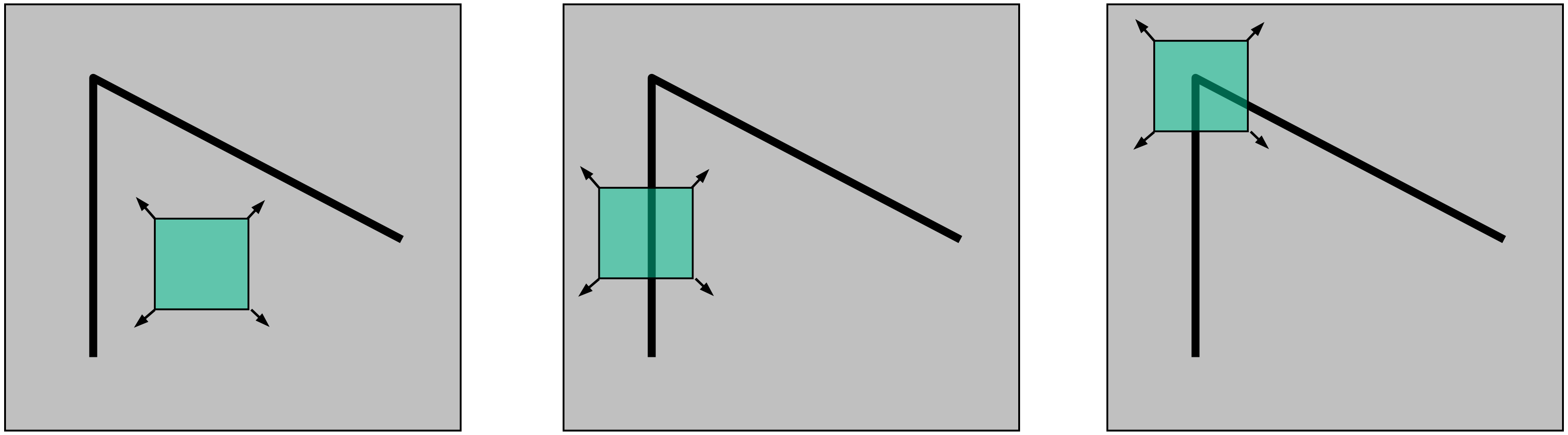


Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Feature detection

Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*



"flat" region:
no change in all directions

"edge":
no change along the edge direction

"corner":
significant change in all directions

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.

Invariance

Suppose you **rotate** the image by some angle

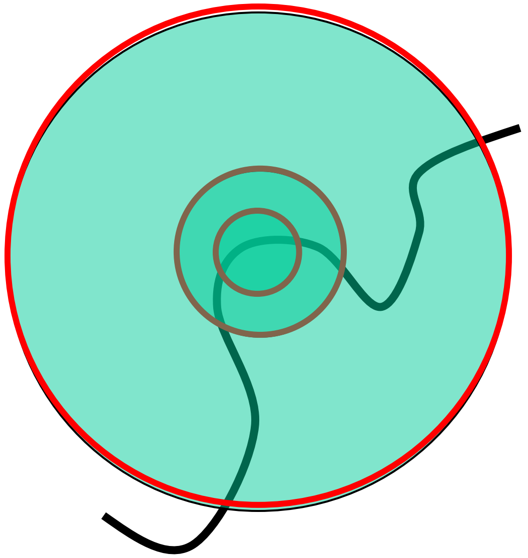
- Will you still pick up the same features?

What if you change the brightness?

Scale?

Scale invariant detection

Suppose you're looking for corners



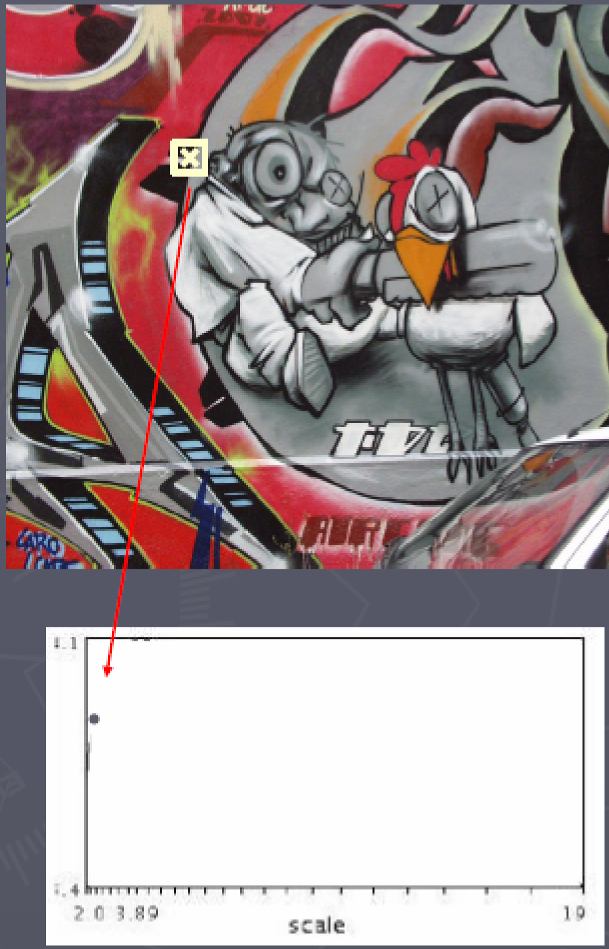
Key idea: find scale that gives local maximum of f

- f is a local maximum in both position and scale
- Common definition of f: Laplacian
  (or difference between two Gaussian filtered images with different sigmas)

Automatic scale selection

Lindeberg et al., 1996

Function responses for increasing scale
Scale trace (signature)



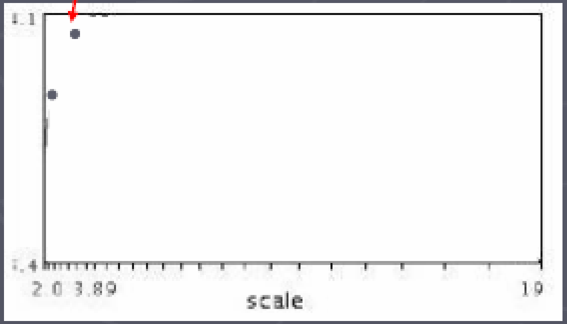
$f(I_{i_1 \ldots i_m}(x,\sigma))$

Slide from Tinne Tuytelaars

Automatic scale selection

Function responses for increasing scale

Scale trace (signature)

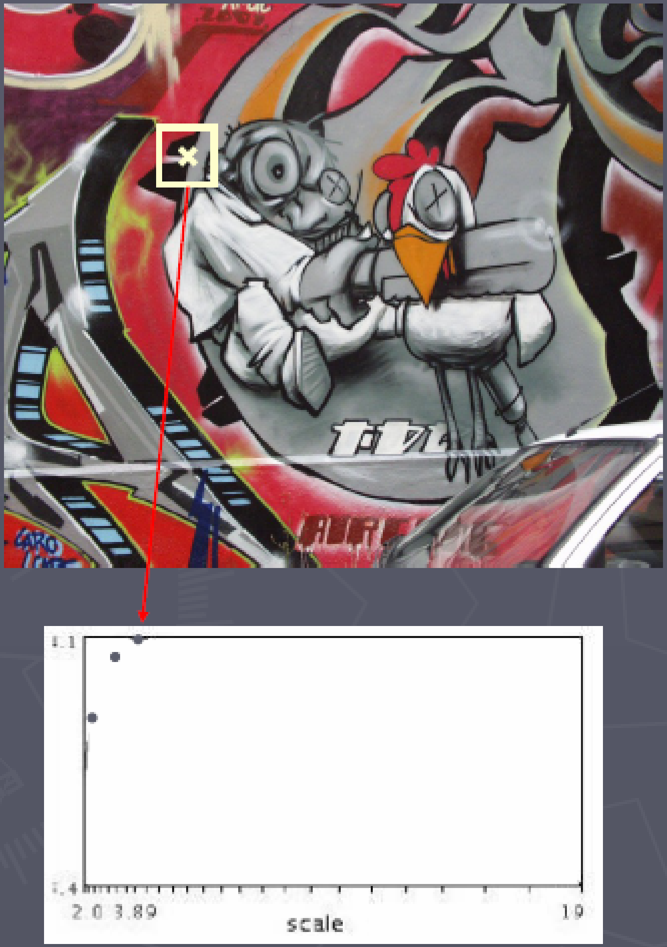


$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection

Function responses for increasing scale

Scale trace (signature)

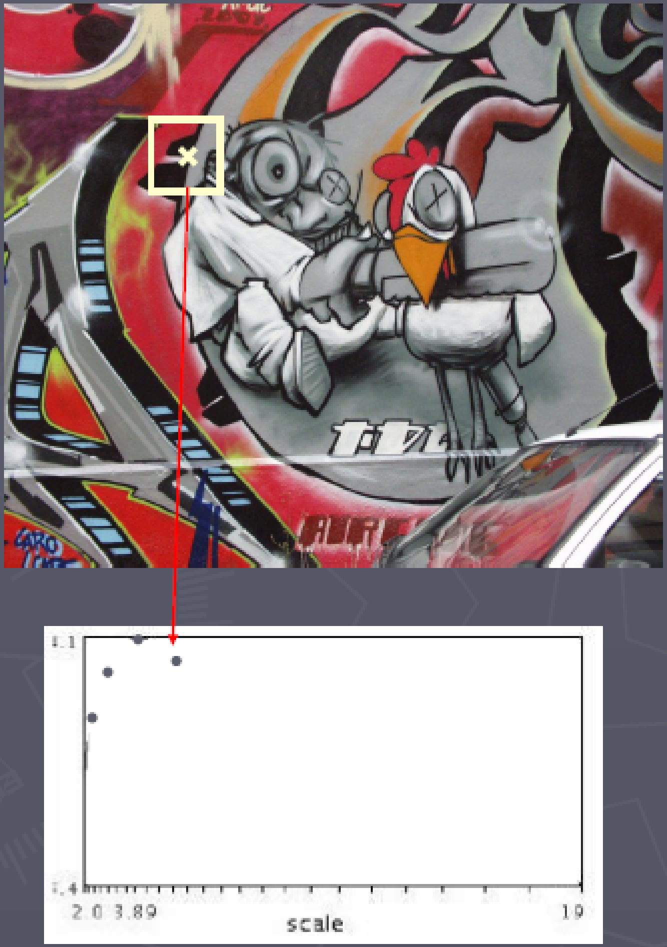


$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection

Function responses for increasing scale

(Scale trace (signature))

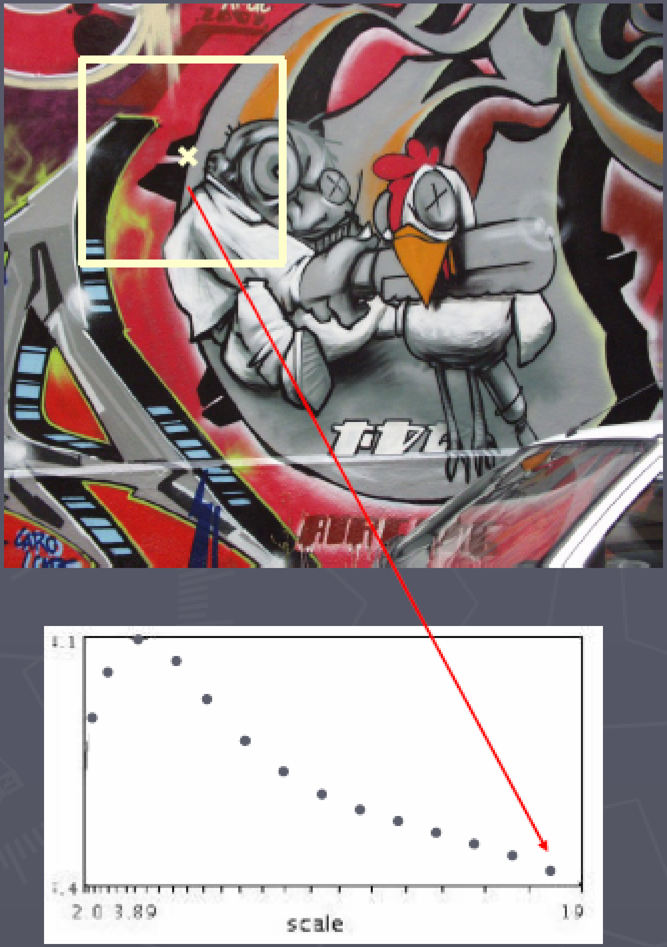


$f(I_{i_1\ldots i_m}(x,\sigma))$

Automatic scale selection

Function responses for increasing scale

Scale trace (signature)

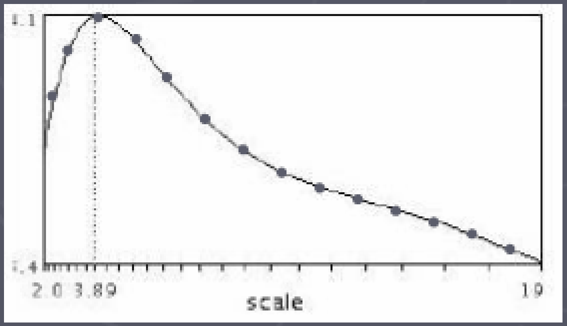


$f(I_{i_1 \ldots i_m}(x,\sigma))$

Automatic scale selection

Function responses for increasing scale

Scale trace (signature)

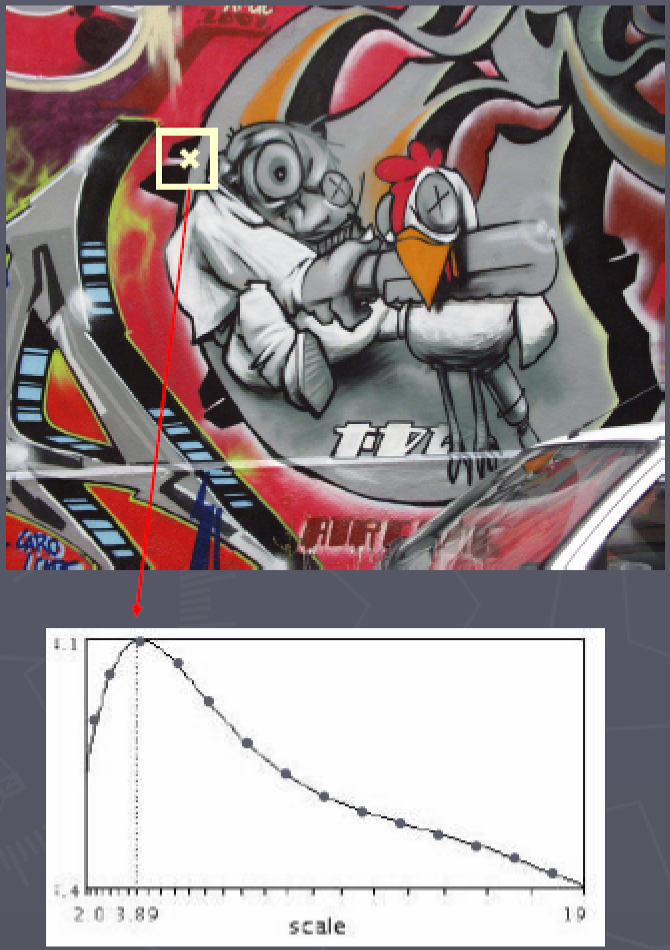


$f(I_{i_1 \ldots i_m}(x,\sigma))$

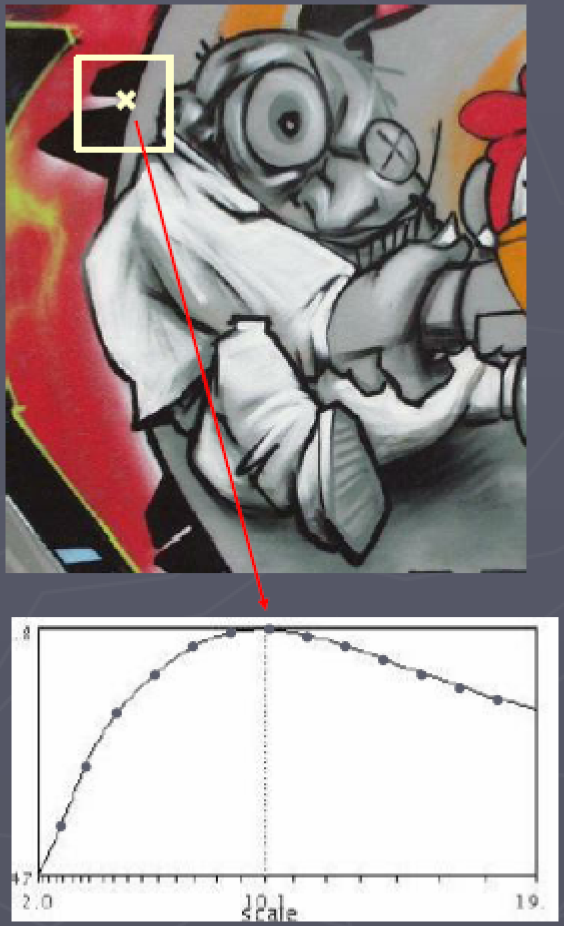
Automatic scale selection

Function responses for increasing scale

Scale trace (signature)



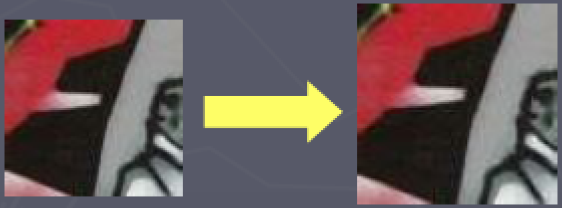
$f(I_{i_1 \dots i_m}(x,\sigma))$

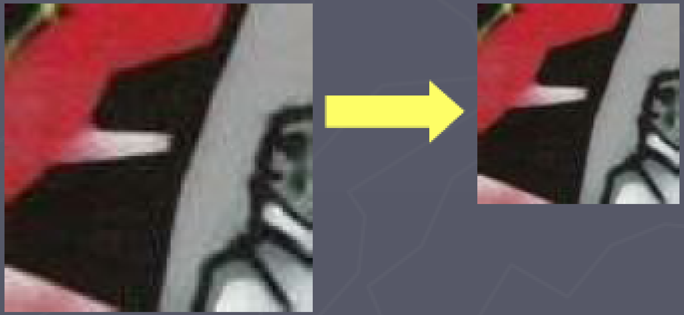


$f(I_{i_1 \dots i_m}(x',\sigma'))$

Automatic scale selection

Normalize: rescale to fixed size





Invariance

Suppose we are comparing two images I_1 and I_2

- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?

We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are 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 transformations (some are fully affine invariant)
  - Limited illumination/contrast changes

How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
    - Harris is invariant to translation and rotation
    - Scale is trickier
        - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
        - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)
2. Design an invariant feature *descriptor*
    - A descriptor captures the information in a region around the detected feature point
    - The simplest descriptor: a square window of pixels
        - What’s this invariant to?
    - Let’s look at some better approaches…

Rotation invariance for feature descriptors

Find dominant orientation of the image patch

- This is given by $\mathbf{x}_+$, the eigenvector of **H** corresponding to λ_+
  - λ_+ is the *larger* eigenvalue
- Rotate the patch according to this angle

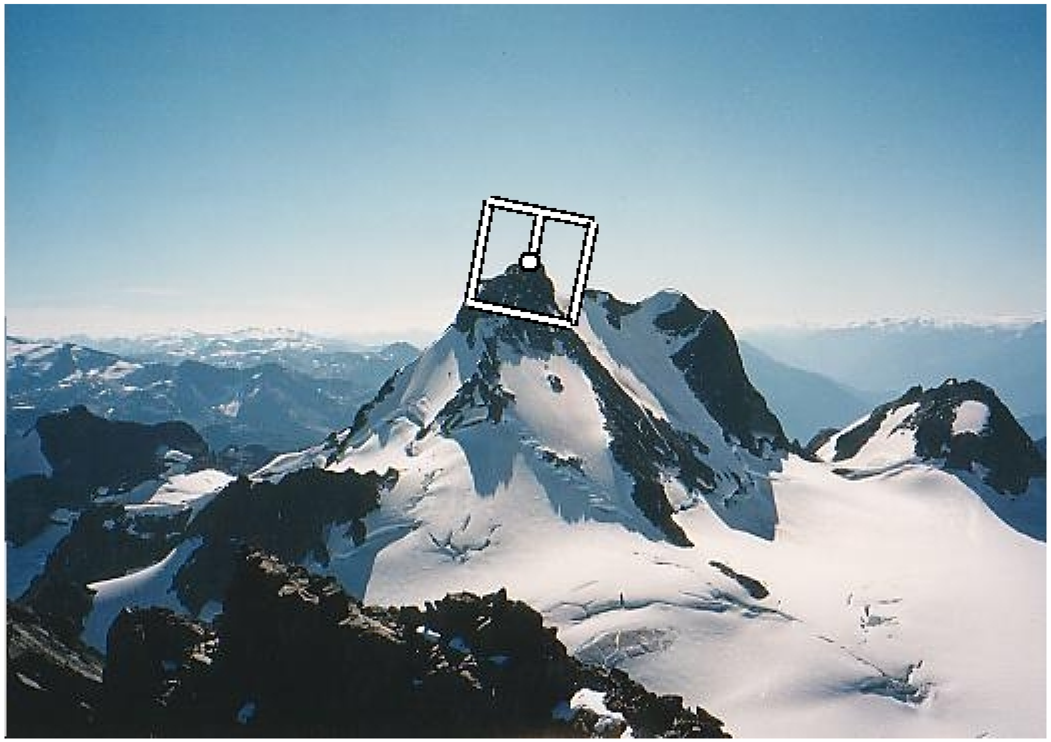
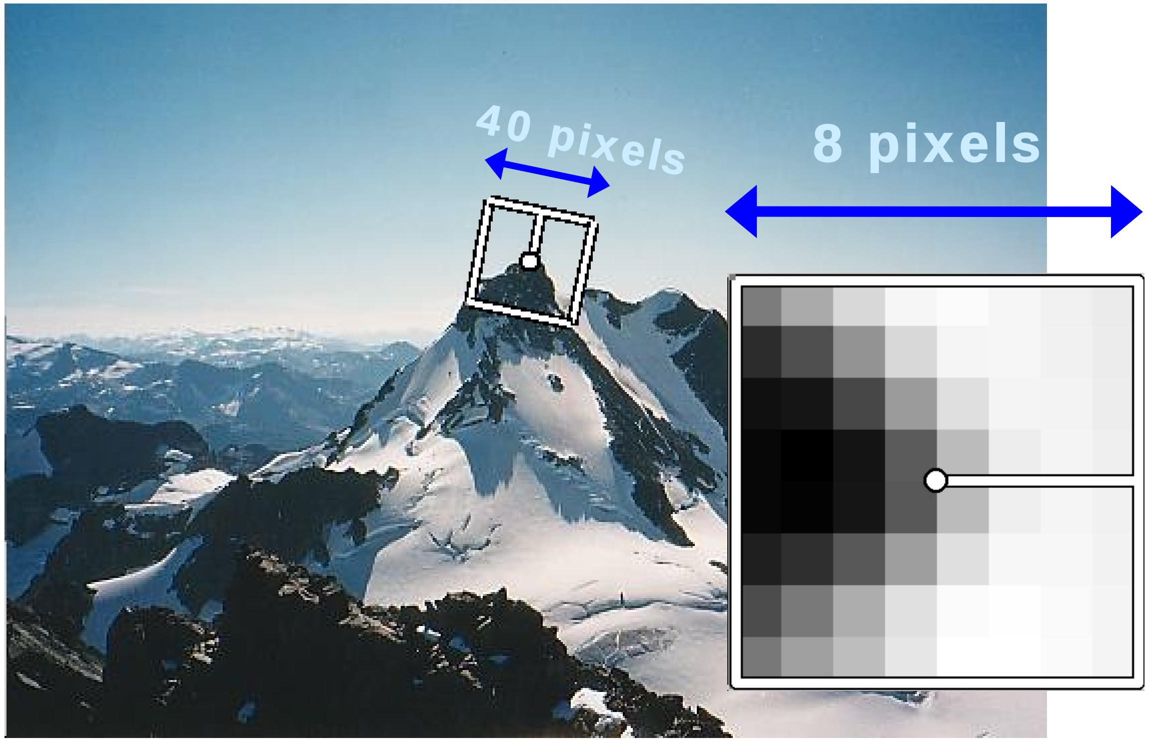


Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Adapted from slide by Matthew Brown

Detections at multiple scales

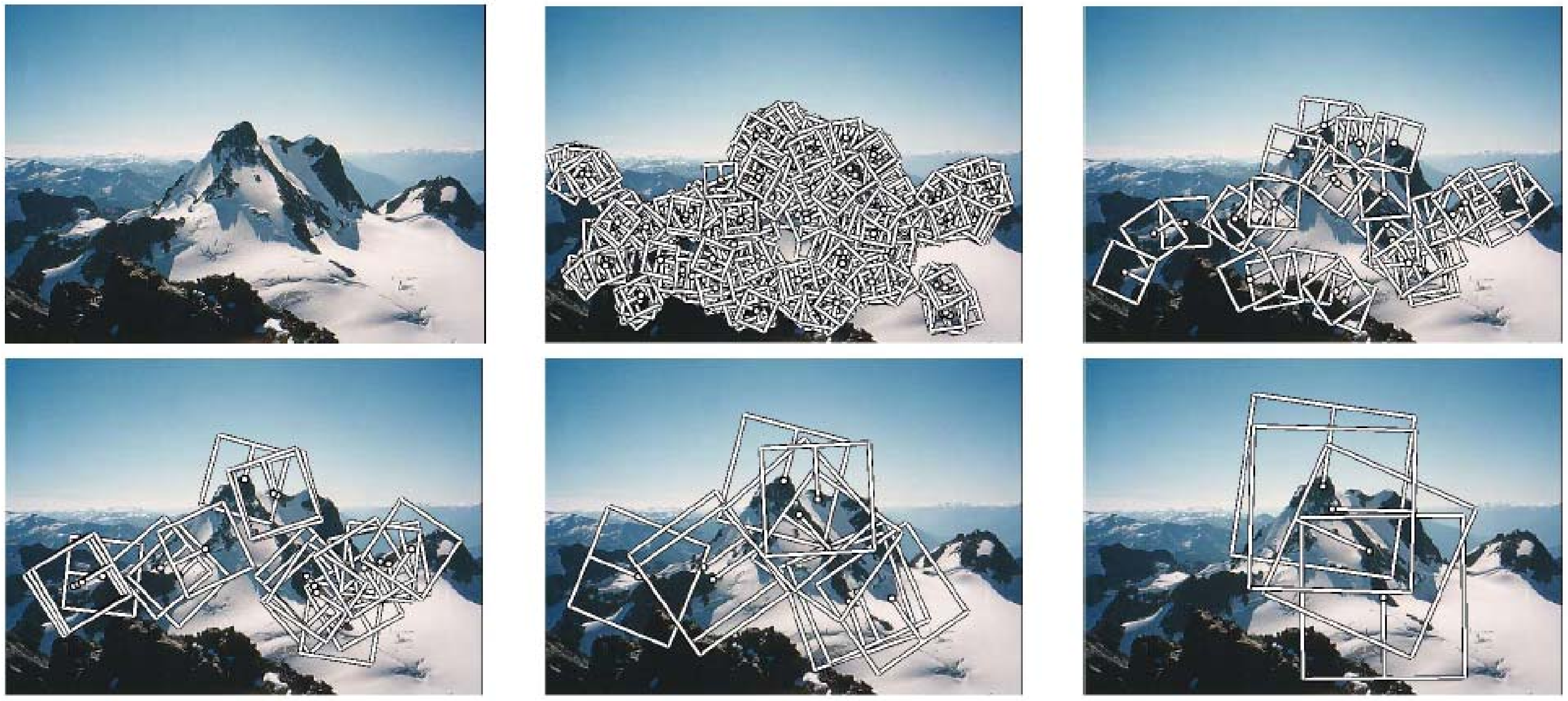
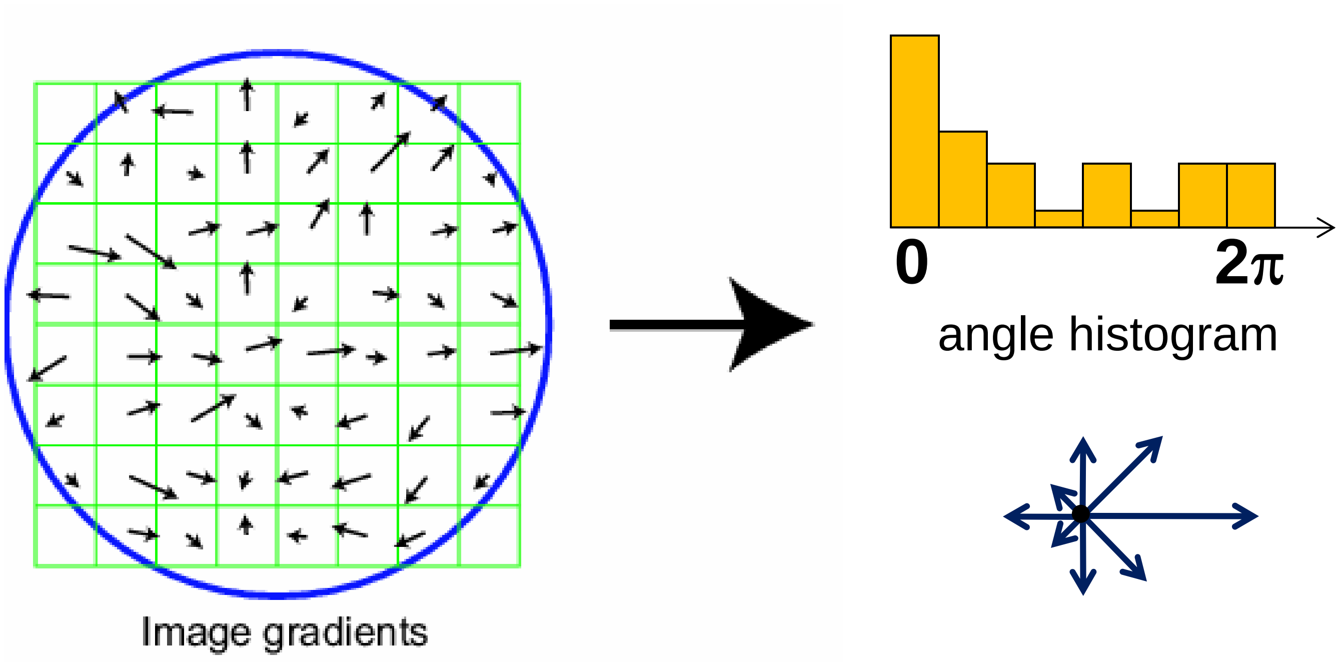


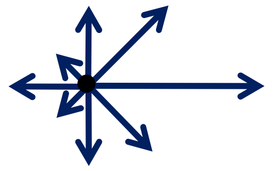
Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

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



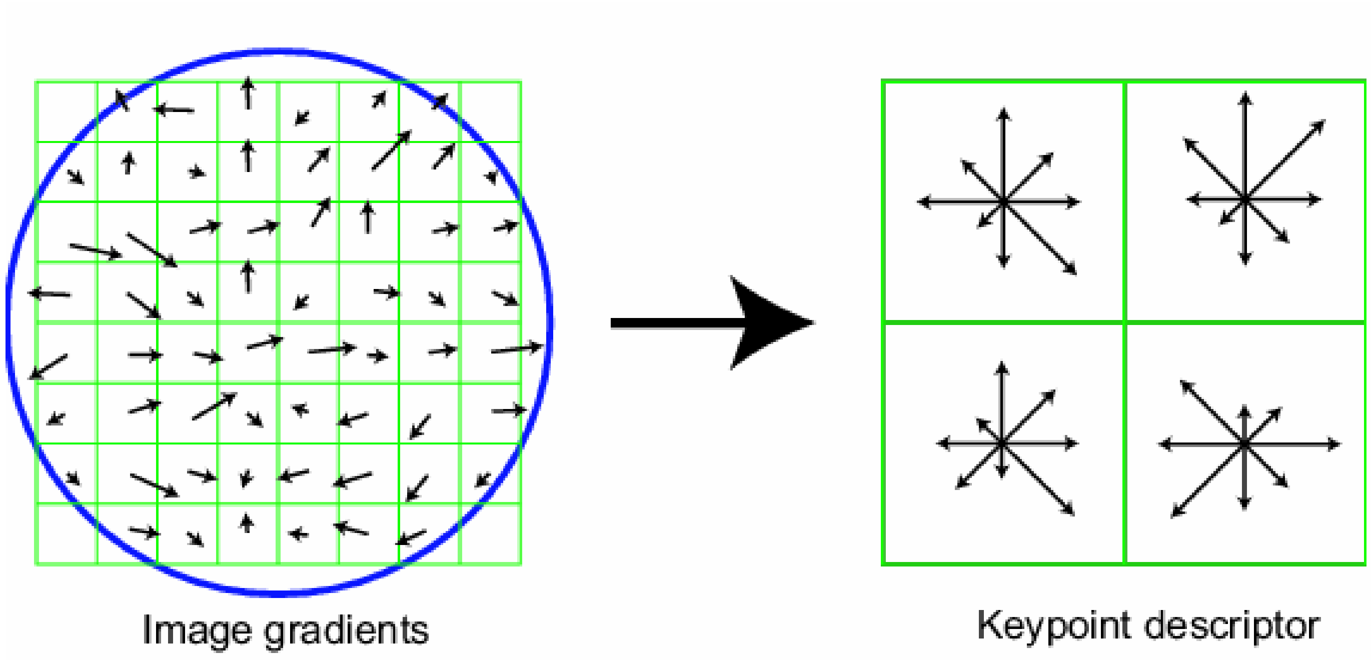


Adapted from slide by David Lowe

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

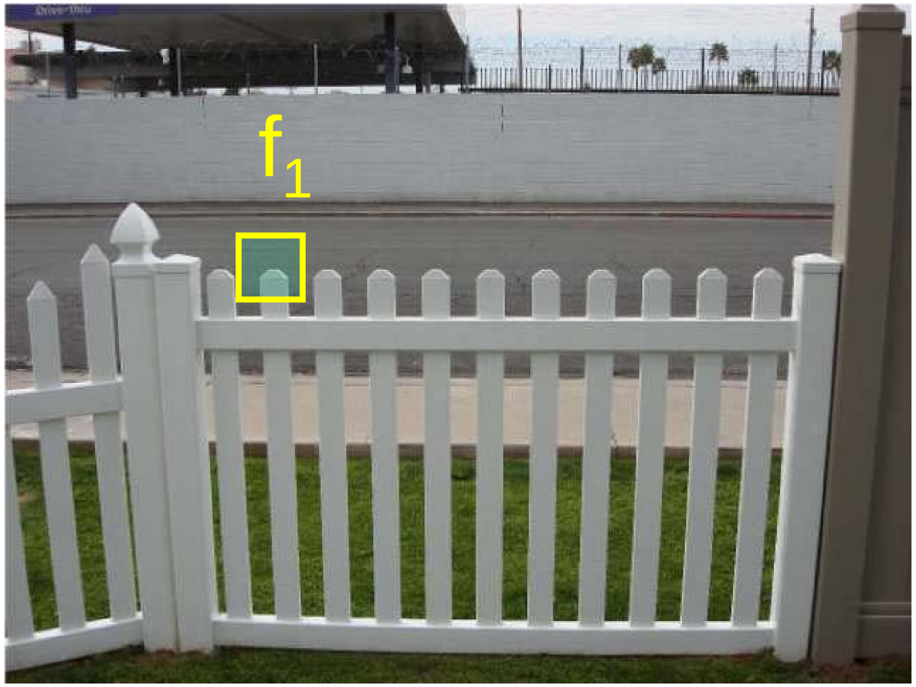


Adapted from slide by David Lowe

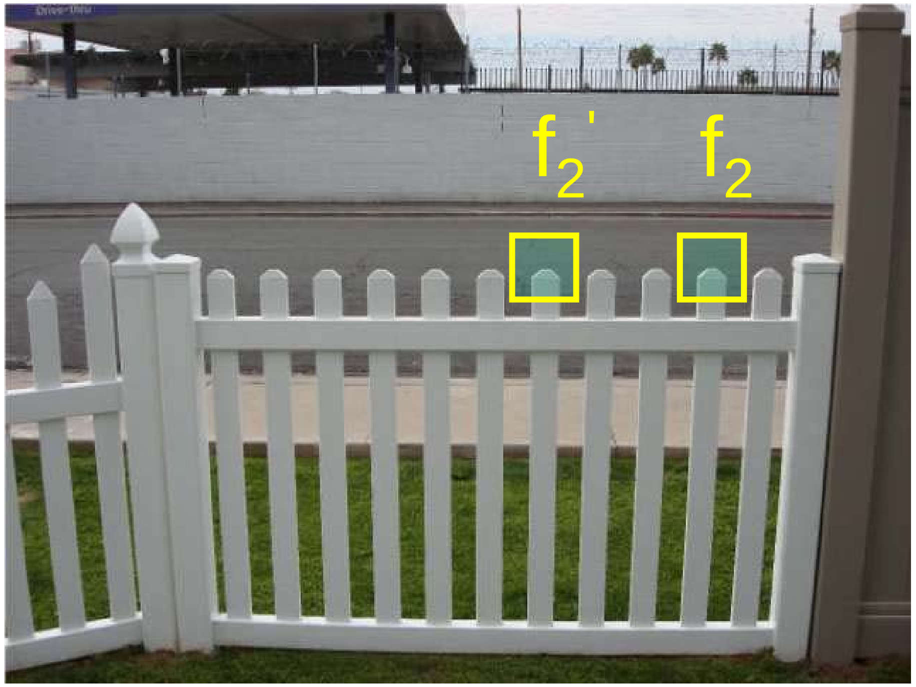
Feature distance

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

- Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
  - 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 small values for ambiguous matches



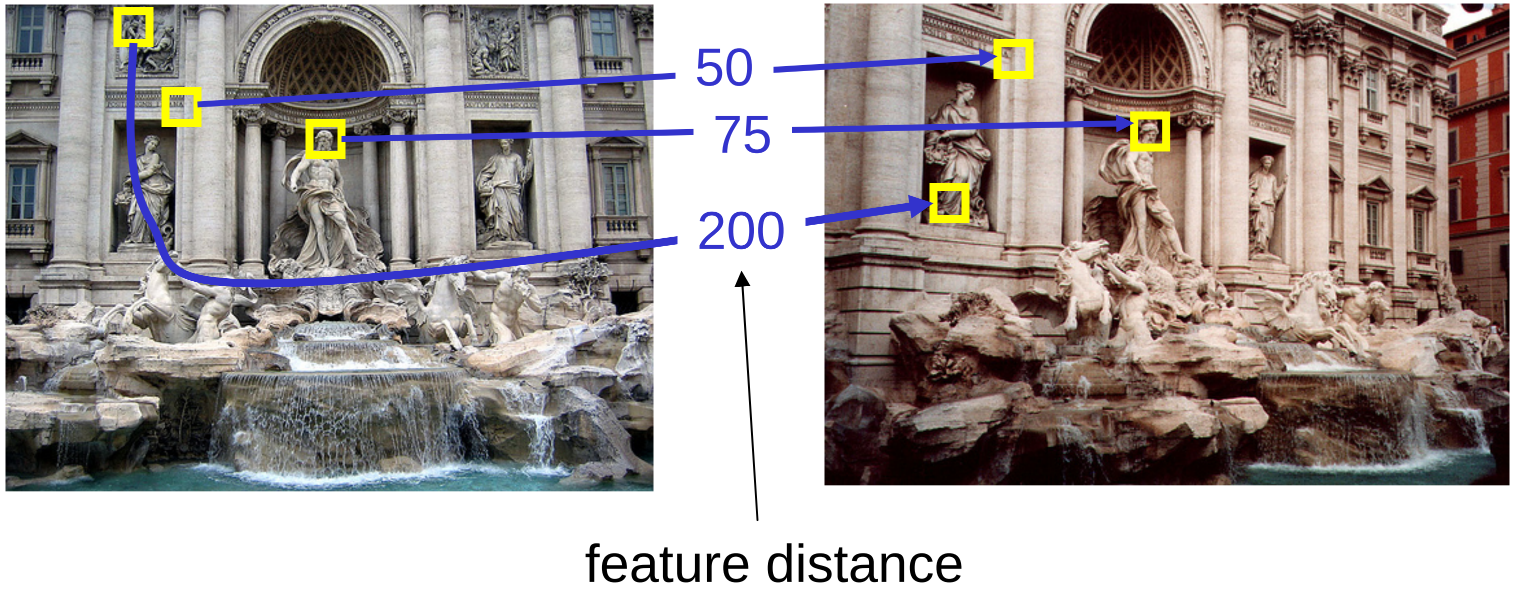
I_1



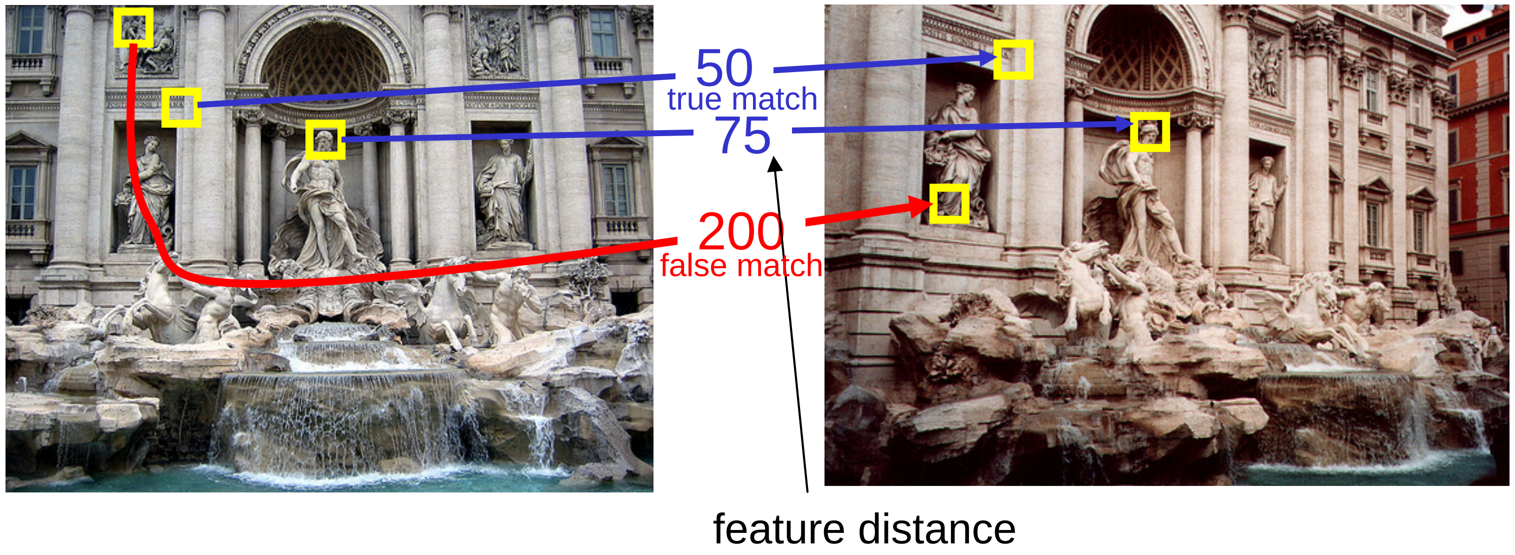
I_2

Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives



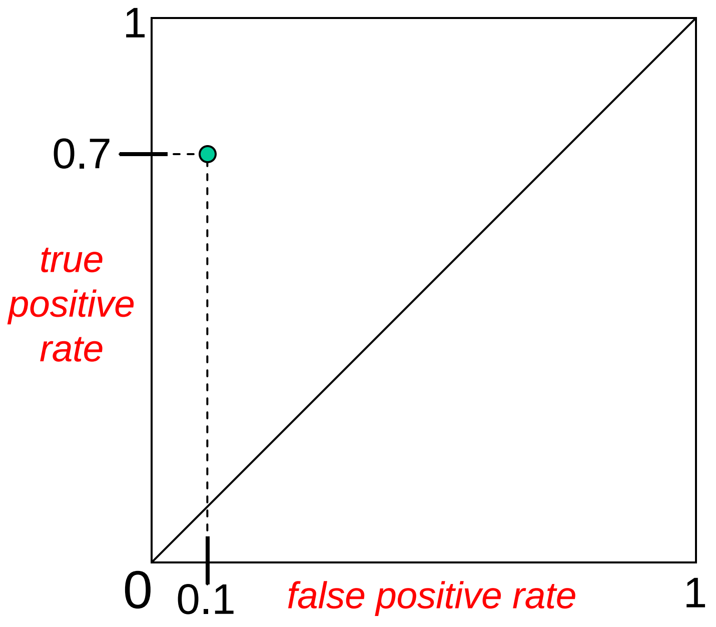
The distance threshold affects performance

- 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?

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

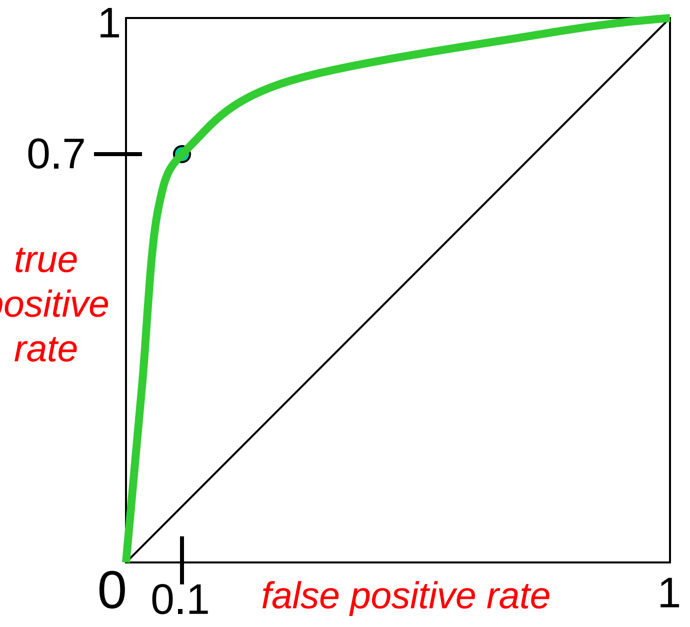


$$\frac{\text{\# false positives}}{\text{\# unmatched features (negatives)}}$$

Evaluating the results

How can we measure the performance of a feature matcher?

ROC curve **("Receiver Operator Characteristic")**



$$\text{true positive rate} = \frac{\text{\# true positives}}{\text{\# matching features (positives)}}$$

$$\text{false positive rate} = \frac{\text{\# false positives}}{\text{\# unmatched features (negatives)}}$$

ROC Curves

- Generated by counting # current/incorrect matches, for different threholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods
- For more info: http://en.wikipedia.org/wiki/Receiver_operating_characteristic