Local Features II: Descriptors

CS 4243 Computer Vision & Pattern Recognition

Angela Yao

Recap & Outline

Last Week

- Keypoints: locations in images for computing descriptors and matching
- Corners & Harris operator
- Equivariance & invariance
- Automatic scale selection
- Laplacian of Gaussian (LoG) operator

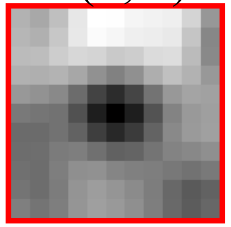
Today's Lecture

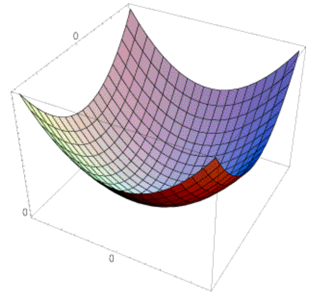
- Definition and characteristics of (ideal) image descriptors
- MOPS, GIST, SIFT
- SIFT as a keypoint detector and descriptor
- Feature matching and evaluation

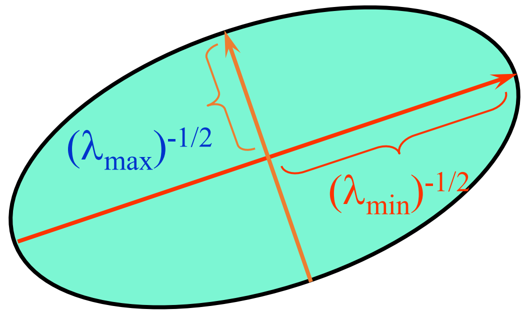
Review: Harris corner detector

- Approximate distinctiveness by "SSD error" (local auto-correlation)
- Approximate SSD with H, the second moment matrix
- Quantify distinctiveness (or cornerness) as function of the eigenvalues of H
- Don't actually need to compute the eigenvalues, just use the determinant and trace of H

$E(u, v)$







Harris & Stephens (1988)

$$R = \det(H) - \kappa \operatorname{trace}^2(H)$$

Harris Corner Detector

1) Compute gradient at each point in image
2) Compute H matrix for a window centered at every pixel in the image

$$H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Same as convolving w/ a Gaussian kernel!

3) Compute *cornerness* score: det(H) – k(tr(H))^2
4) Find points whose surrounding window gave large corner response (R > threshold)
5) Take the points of local maxima, i.e., perform non-maximum suppression

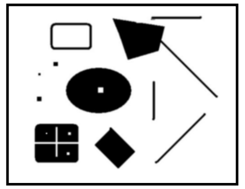
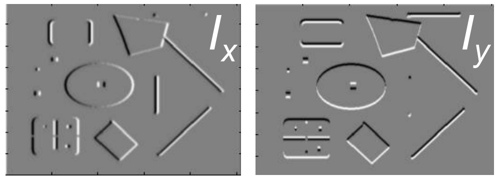
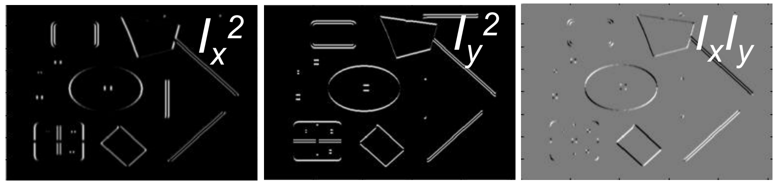


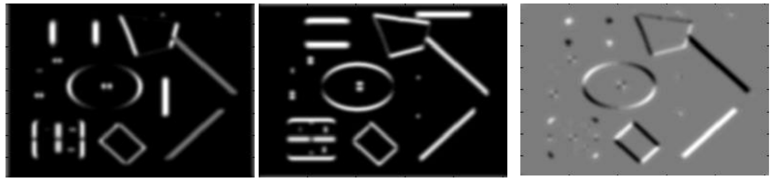
image gradients

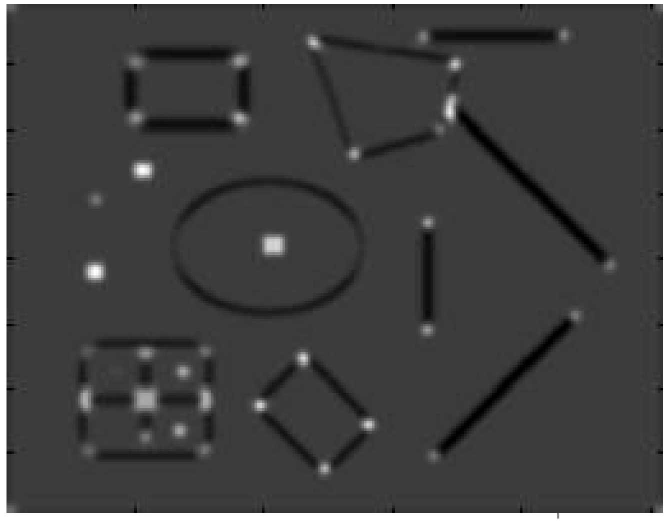


square of gradients



convolve w/ Gaussian



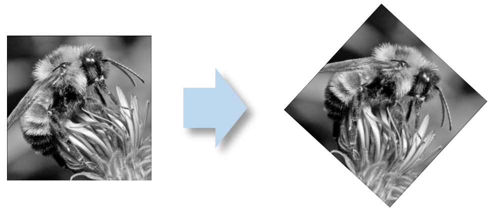


C.Harris and M.Stephens. [“A Combined Corner and Edge Detector.”] *Proceedings of the 4th Alvey Vision Conference.* 1988.

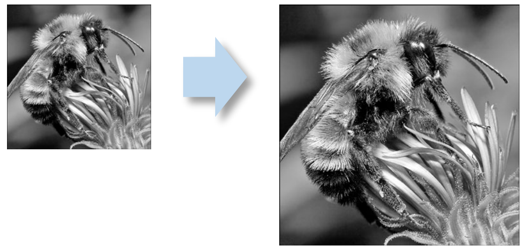
Corner locations are

Geometric transformations

Equivariant to **Rotation** and translation

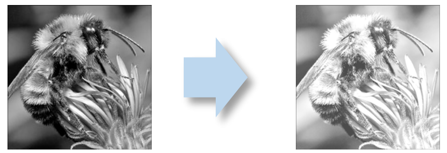


NOT equivariant to **Scale**



Photometric transformations

(semi-)invariant to **Intensity change**



Invariant to constant changes but not to scaling changes

Adapted from K. Grauman

Recall: How can we combine two images?

We've actually already looked at some descriptors in class...

- Colour/texton histogram
- Mean gradients
- Filter bank response

1. Match parts which are the same on both images

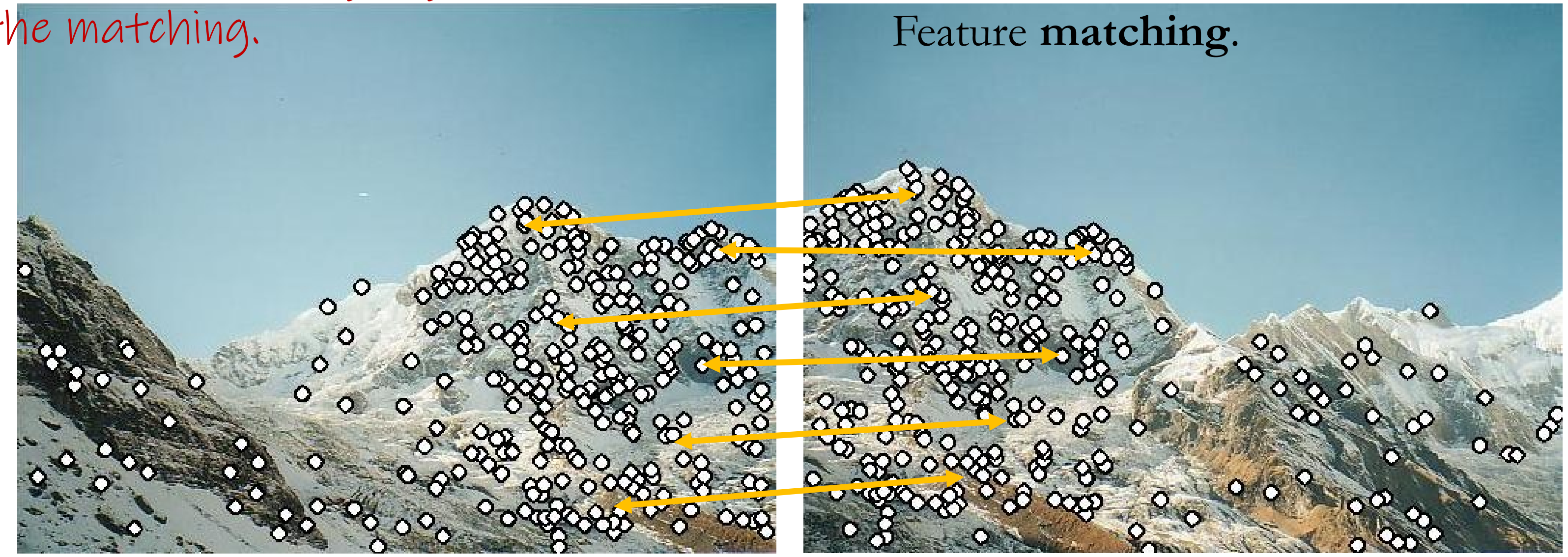
   a. Find locations to match. — Interest point **detection**.

   b. Represent surrounding region mathematically. — Compute feature **descriptors**.

   c. Do the matching. — Feature **matching**.

Last lecture

This lecture



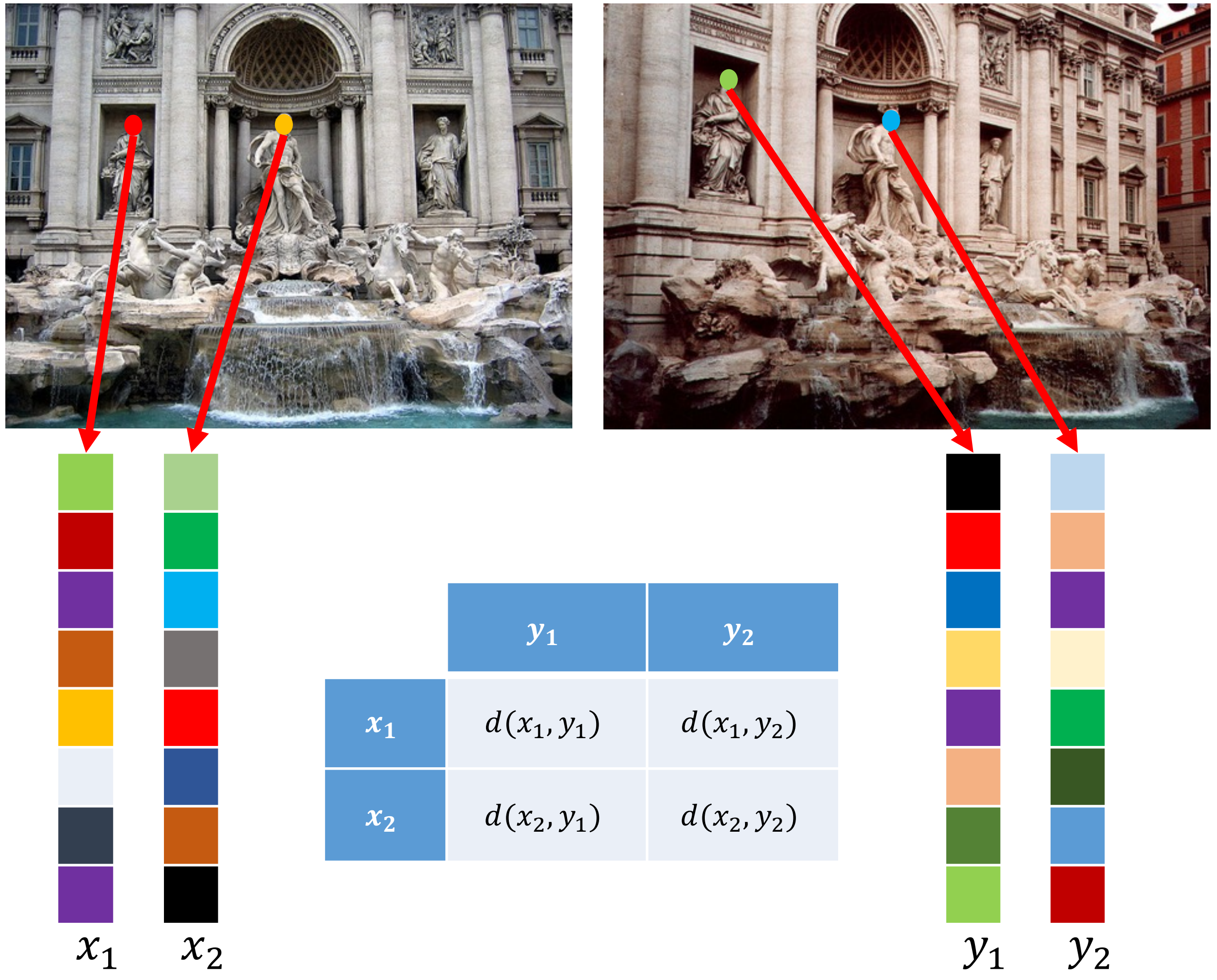
Sometimes, the matching process is also called registration.

Adapted from K. Grauman, S. Lazebnik

Image Descriptors

Invariance, Histogramming
MOPS, GIST

Definition of Image Feature Descriptors



	y_1	y_2
x_1	$d(x_1, y_1)$	$d(x_1, y_2)$
x_2	$d(x_2, y_1)$	$d(x_2, y_2)$

- Descriptors are vector representations that mathematically characterize a region in the image.
- For matching, we want to measure the distance (or similarity) between every pair of descriptors.
- Descriptors should be:

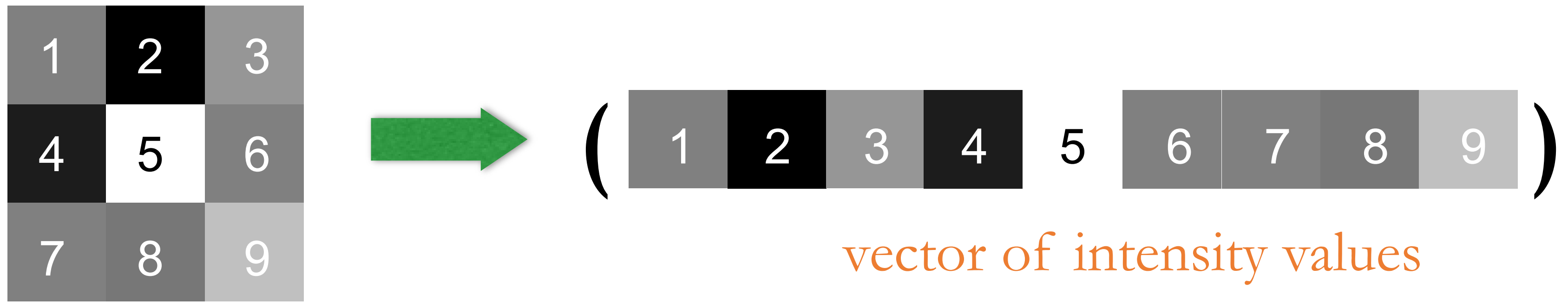
  Invariant / equivariant: shouldn't change even if image is transformed (geometric, photometric)

  Discriminative: (sufficiently) unique for each point

Adapated from K. Kitani

Raw Image Patch: Intensity

Just use the pixel values of the patch



vector of intensity values

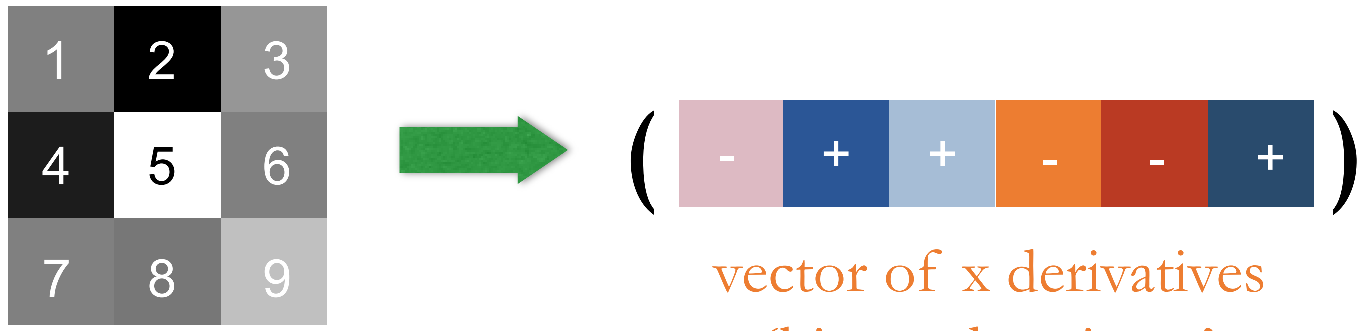
Perfectly fine if geometry and appearance is unchanged
(a.k.a. template matching)

How can we be less sensitive to absolute intensity values?

Slide credit: K. Kitani

Image Gradients

Use pixel differences, i.e. gradients



vector of x derivatives
'binary descriptor'

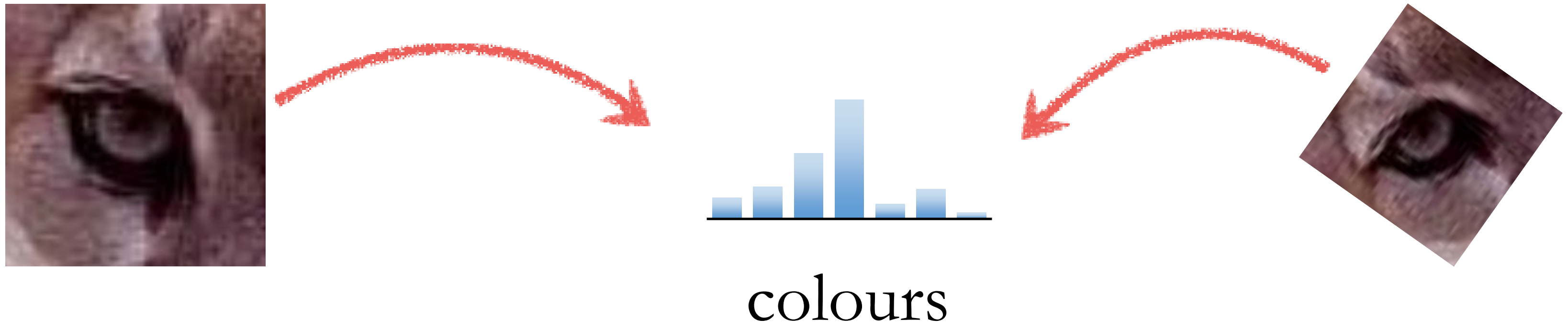
Descriptor is invariant to absolute intensity values

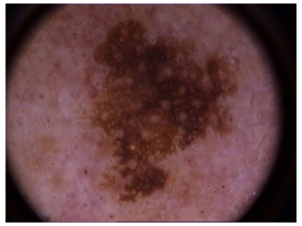
How can we be less sensitive to deformations?

Slide credit: K. Kitani

Colour Histogram

Count the colors in the image using a histogram.





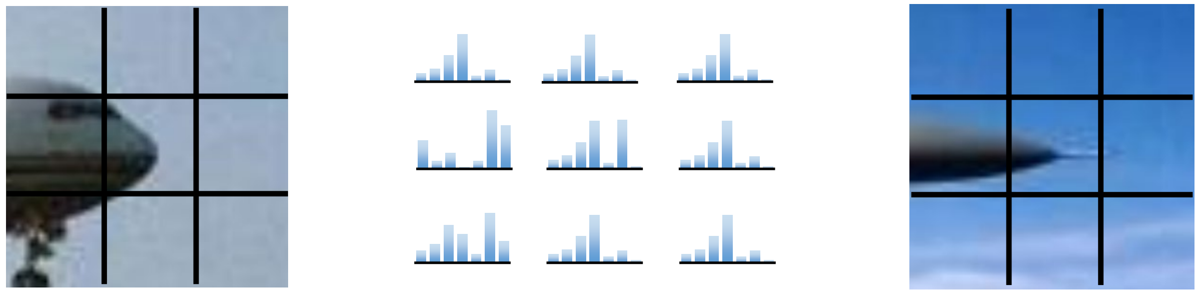
Invariant to changes in scale and rotation

How can we introduce more sensitivity to the spatial layout?

Slide credit: K. Kitani

Spatial Histograms

Compute histograms over spatial ‘cells’



Retains rough spatial layout

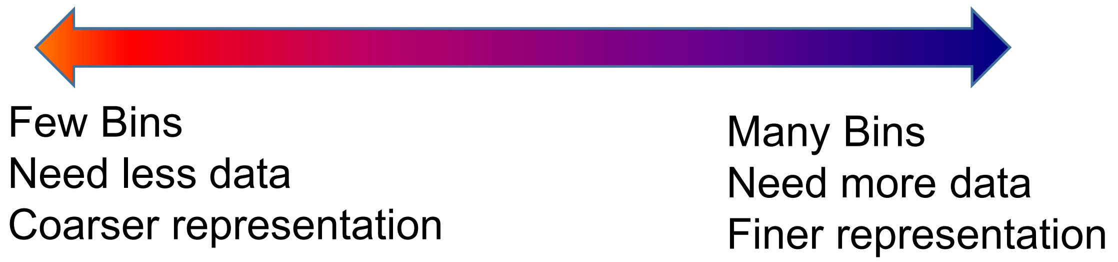
Some invariance to deformations

How can we be completely invariant to rotation?

Slide credit: K. Kitani

An Aside on Histograms

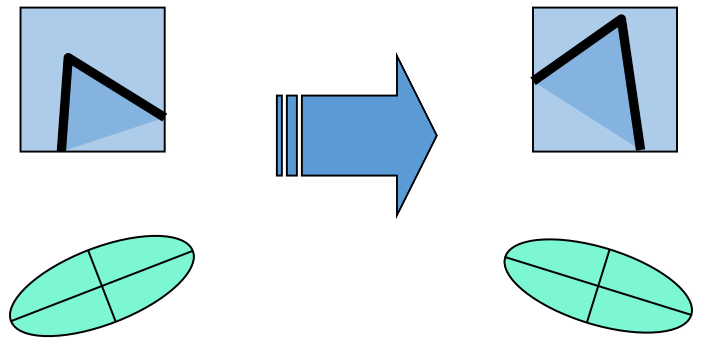
- What should we be histogramming and how?
  - Grids of entire domain? Fast but applicable only with when dimensionality is low
  - Clustering: slower but can quantize data in higher dimensions
  - Example: greyscale intensity, from 0-255 uniformly? Cluster first based on data occurrence?
- Where in the image should we get these values?
- Histogram bin sizes?



- How to compare histograms?
  - Histogram intersection or Euclidean may be faster
  - Chi-squared often works better

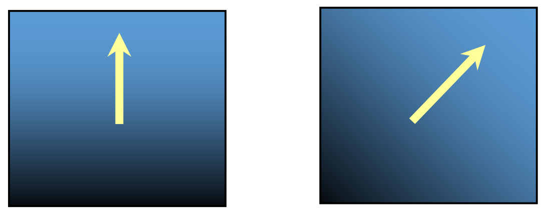
Rotation Invariant Descriptors

- Harris corner response measure:
  depends only on the eigenvalues of the matrix *H,* so the process of *finding* the keypoint is equivariant to rotations.



- Find local orientation:
  Dominant gradient direction for image patch

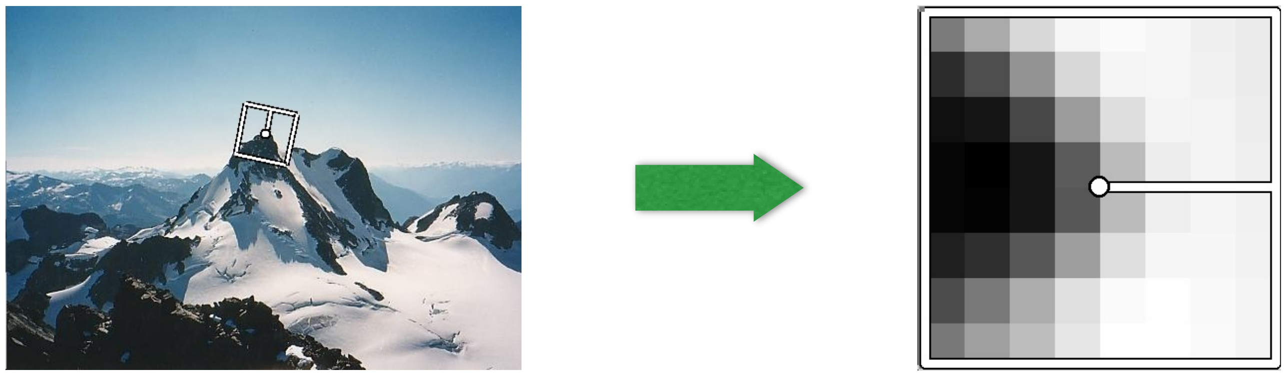
- Rotate patch according to this angle:
  Puts the patches into a canonical orientation.



Adapted from: S. Seitz, D. Frolova, D. Simakov

Orientation Normalization

Normalize the orientation of the patch according to the dominant image gradient direction.



save the orientation angle θ along with (x, y, s)

What are possible ways to determine the dominant gradient direction of an image patch?

Average the orientations derived from the gradients in region around a keypoint?

Averaging may overly smooth out signal, get wrong gradient direction.

Alternative: mode

Fig. Matthew Brown

Multi-Scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) 2005. [pdf]

Take 40x40 window around detected keypoint (e.g. Harris corner)

absorbs localization errors

Subsample every 5th pixel.

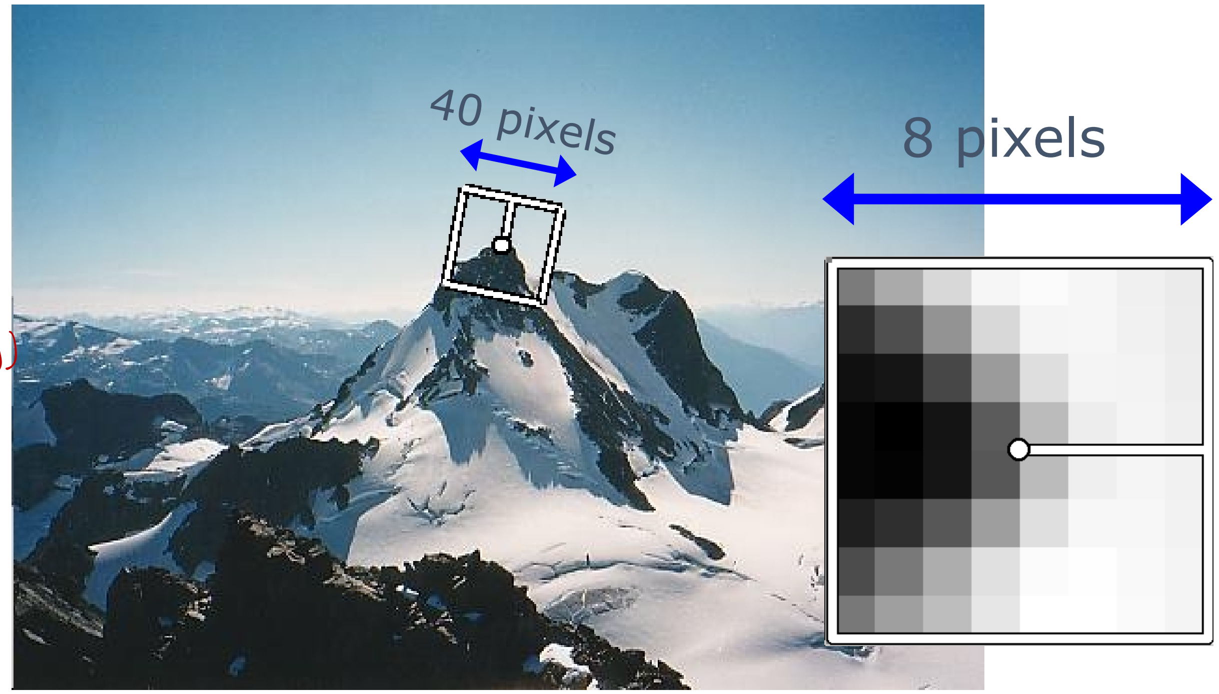
Rotate to horizontal. (virtually via sampling)

Normalize intensity values within the window by subtracting the mean & dividing by the standard deviation.

robustness to photometric variations

Wavelet transform on 8x8 patch to get a 64-dimensional descriptor

Beyond our scope, projection similar to e.g. PCA



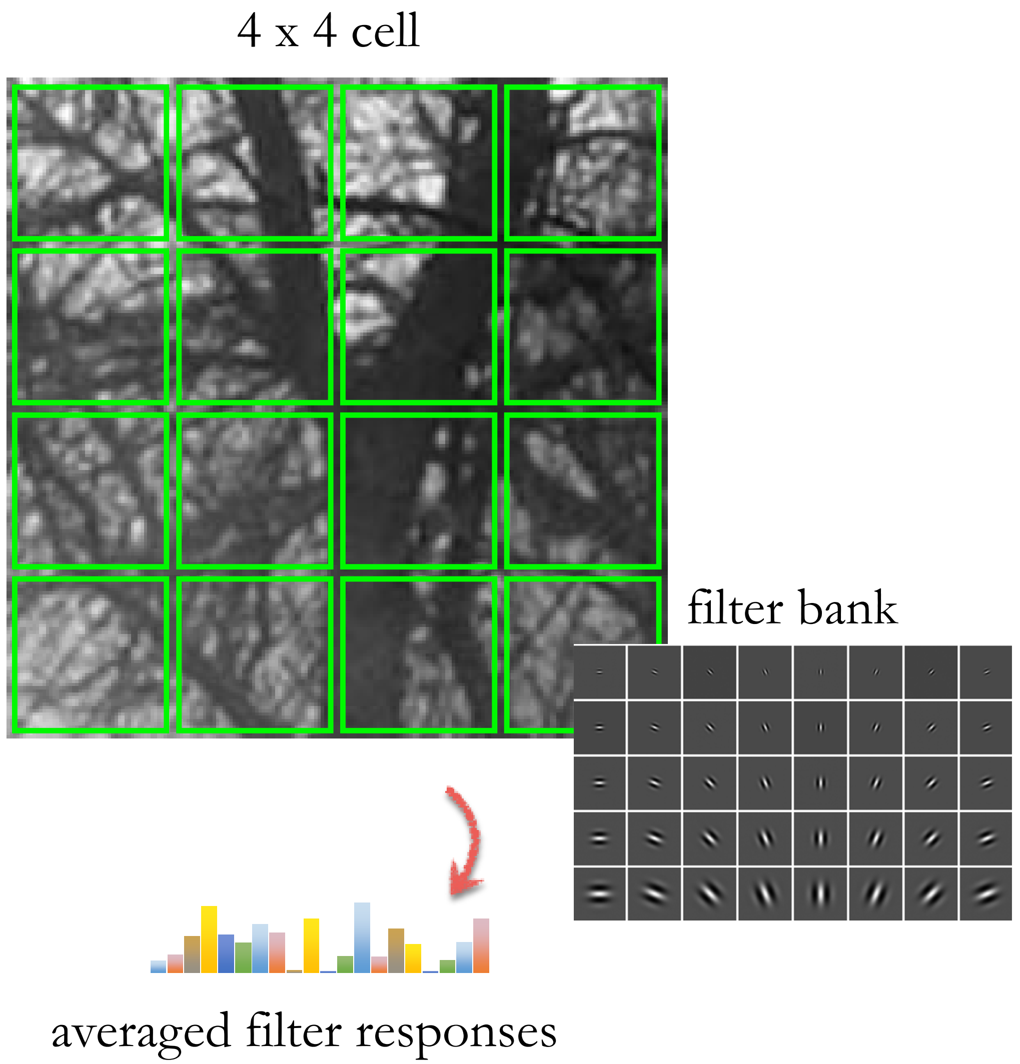
Adapted from Matthew Brown

GIST Descriptor

1. Divide image (patch) into 4 x 4 cells
2. Apply Gabor filters (filter bank of directional edge detectors).
3. Compute filter response averages for each cell.
4. Size of descriptor is 4 x 4 x N, where N is size of the filter bank.

What is being encoded by the GIST descriptor then?

Rough spatial distribution of the image gradients!



Slide credit: K. Kitani

SIFT Feature Descriptors

Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT)

Original SIFT describes both a **detector** and **descriptor**

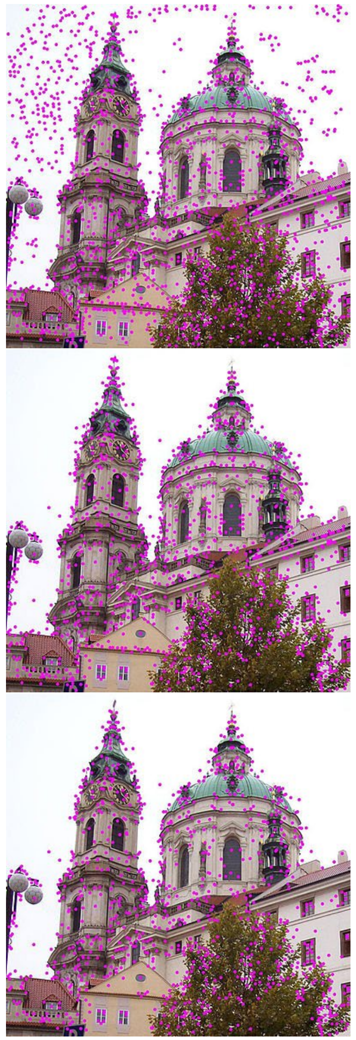
scale-invariance

1. Multi-scale extrema detection
2. Keypoint localization
3. Orientation assignment / rotation
4. Keypoint descriptor

rotation-invariance

Based on the multi-scale "blob" detection (Laplacian of Gaussian), with thresholding to discard low-contrast and low-curvature keypoints (i.e. non-corners)

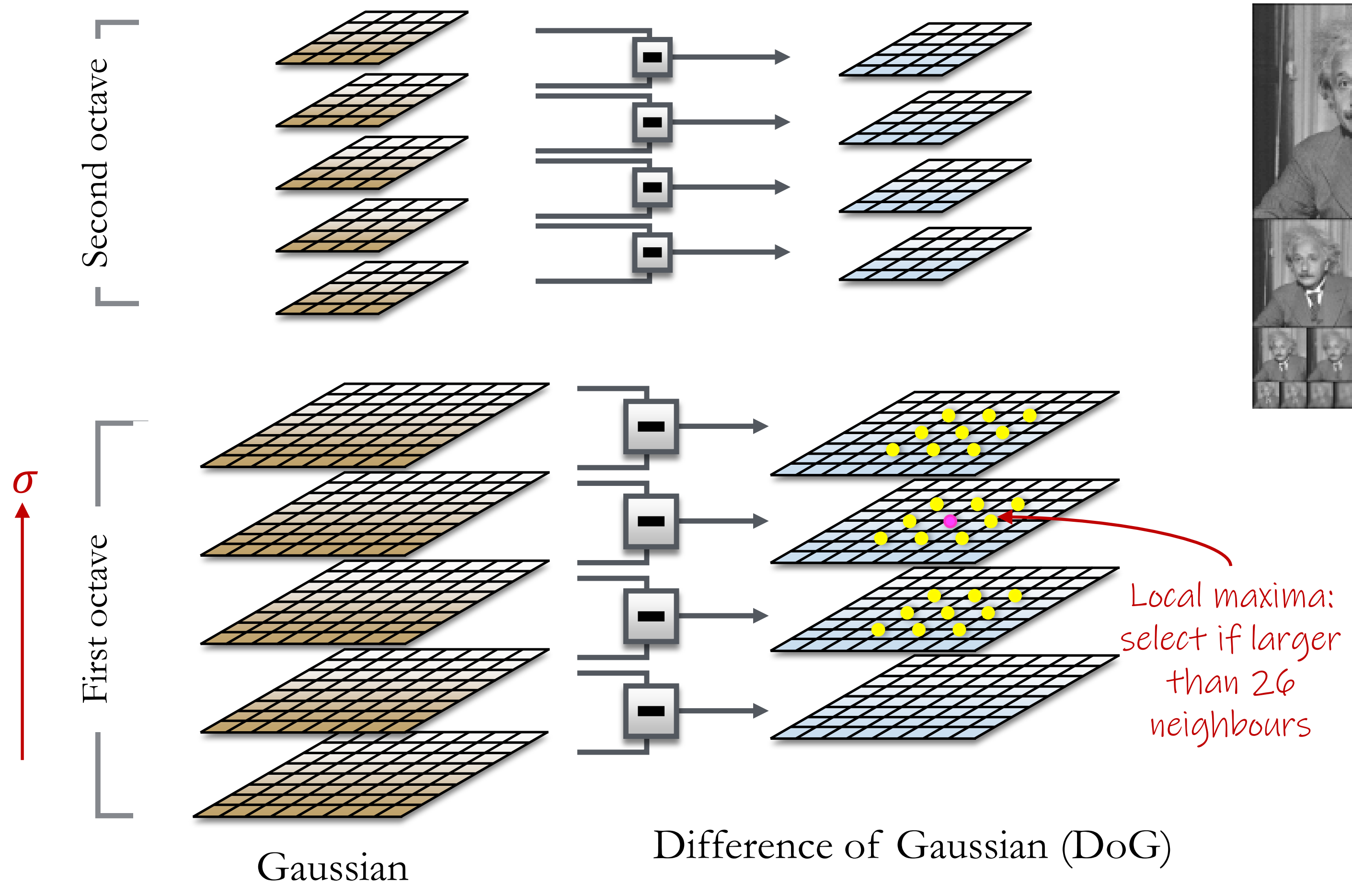
SIFT Descriptor can also be used stand-alone e.g. w/ Harris corners



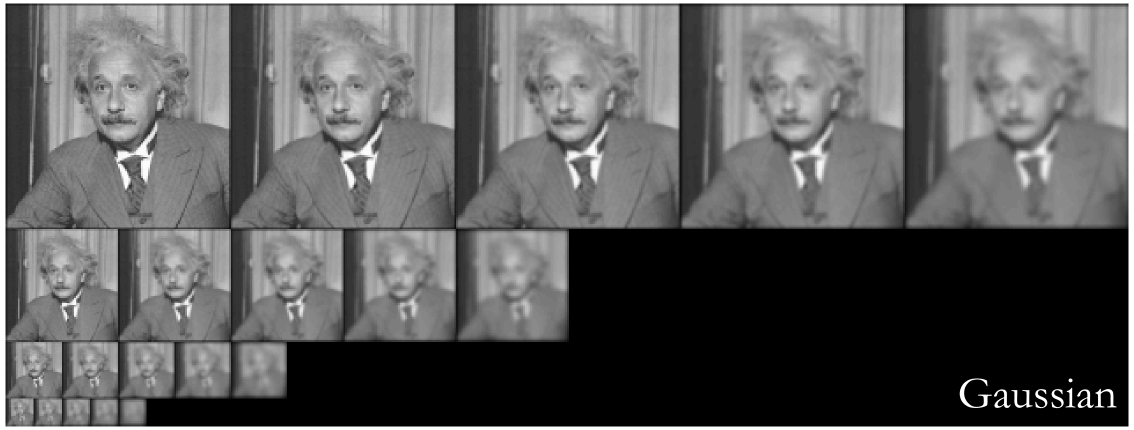
D. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

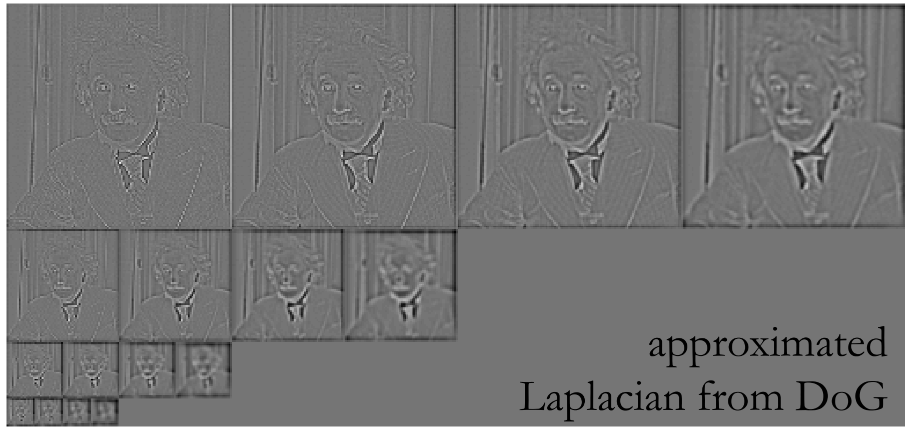
Adapted from K. Kitani;

1. Multi-Scale Extrema (Blob) Detection


Second octave
First octave
σ
Local maxima: select if larger than 26 neighbours
Gaussian
Difference of Gaussian (DoG)

approximation of Laplacian of Gaussians





(2 + 3) Localization + Scale/Orientation

Beyond our scope

In theory, this can be done before searching for local maxima, i.e. thresholding the DoG response image if we do not do any refinement

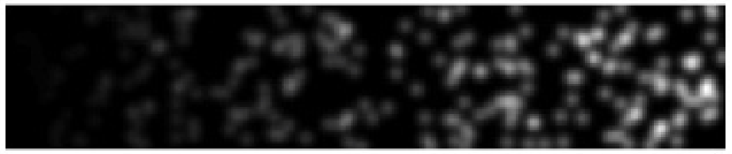
Compute the Hessian(!) of the DoG response image, retain maxima only if ratio of max to min eigenvalue of the Hessian is above some threshold (same as Harris Corner)

See slide 15/16

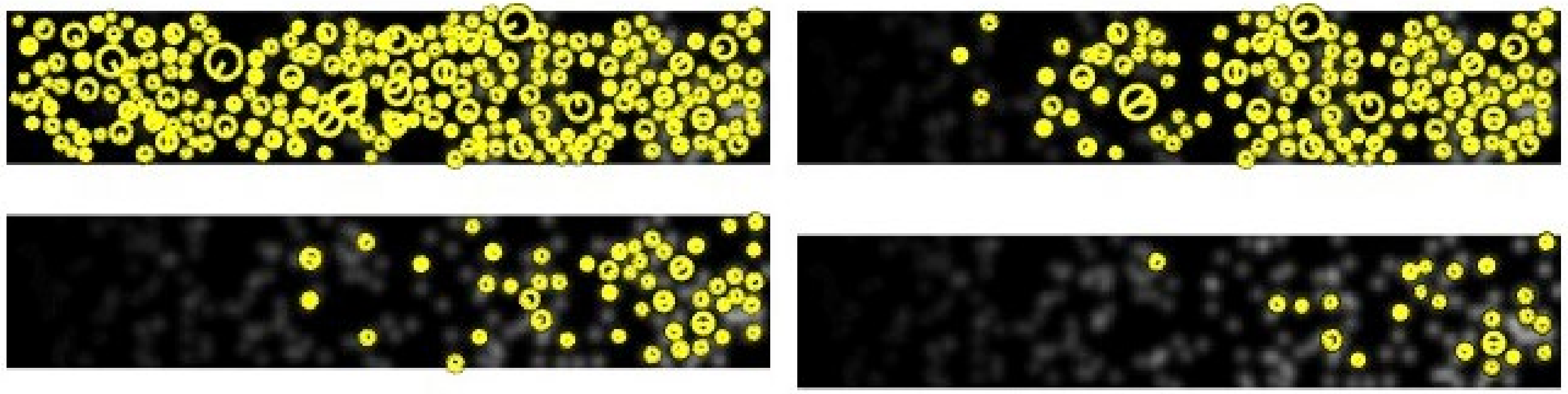
- Refine location of keypoints to sub-pixel accuracy via Taylor series expansion across the octaves
- Reject maxima which do not exceed some threshold
- Reject maxima where the local curvature is insufficient

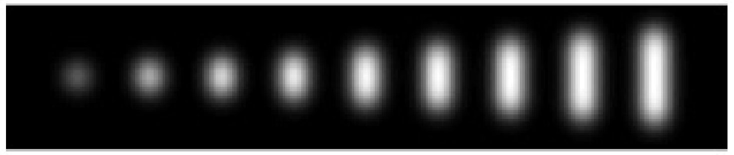
- **Scale**: based on the σ and octave of maxima
- **Orientation**: dominant orientation of gradients

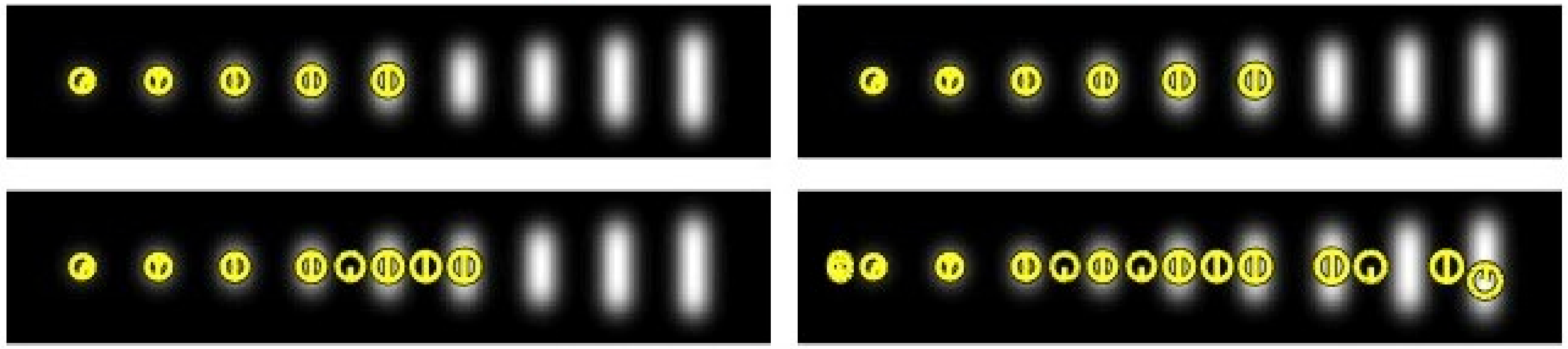
SIFT Keypoint Parameters


A test image for the peak threshold parameter.

Weaker keypoints are removed as threshold increases. Weaker keypoints are found in parts of the image with lower contrast.


Detected frames for increasing peak threshold.
From top: `peak_thresh = {0, 10, 20, 30}`.


A test image for the edge threshold parameter.


Detected frames for increasing edge threshold.
From top: `edge_thresh = {3.5, 5, 7.5, 10}`

http://www.vlfeat.org/overview/sift.html

SIFT Descriptor

1. Take 16x16 window around detected keypoint at from image at scale matching to keypoint. Partition window into a 4x4 grid of cells (gives some sensitivity to spatial layout).
2. Compute gradient orientations and magnitudes for each pixel; reweight magnitudes according to a Gaussian centered on keypoint and discard pixels with low magnitude.
3. Of the remaining edge orientations, create a histogram with 8 orientation bins for each cell. To be rotation-invariant, "shift" histogram binning by the dominant orientation. Collapse into vector (16 x 8 = 128 dims).
4. Normalize vector to unit length, clamp values based on threshold, re-normalize again for final descriptor.

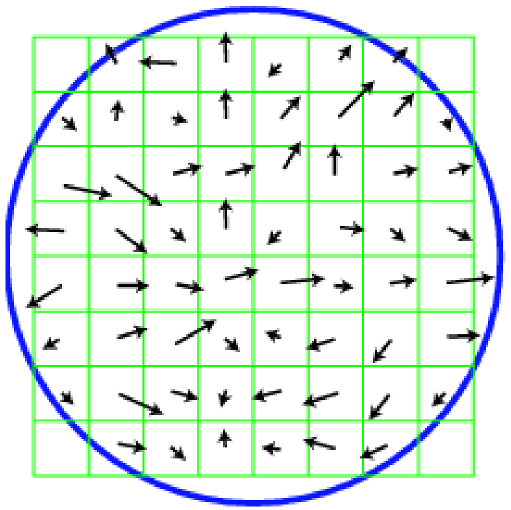
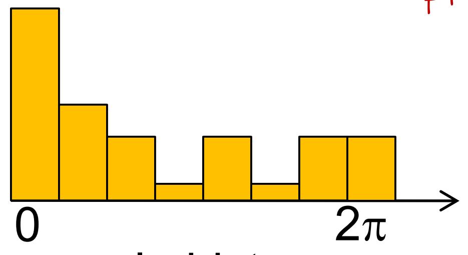
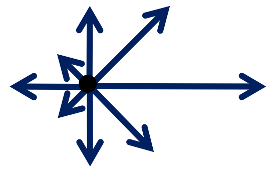


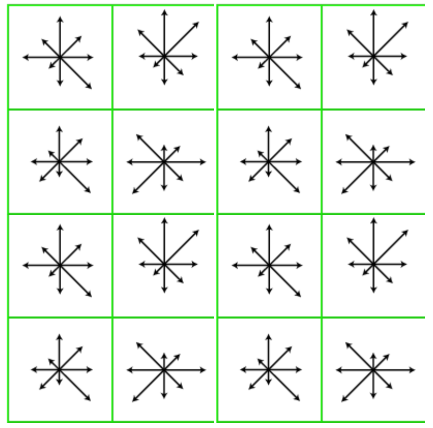
Image gradients



angle histogram

Another visualization of the angle histogram; vector magnitudes proportional to counts.



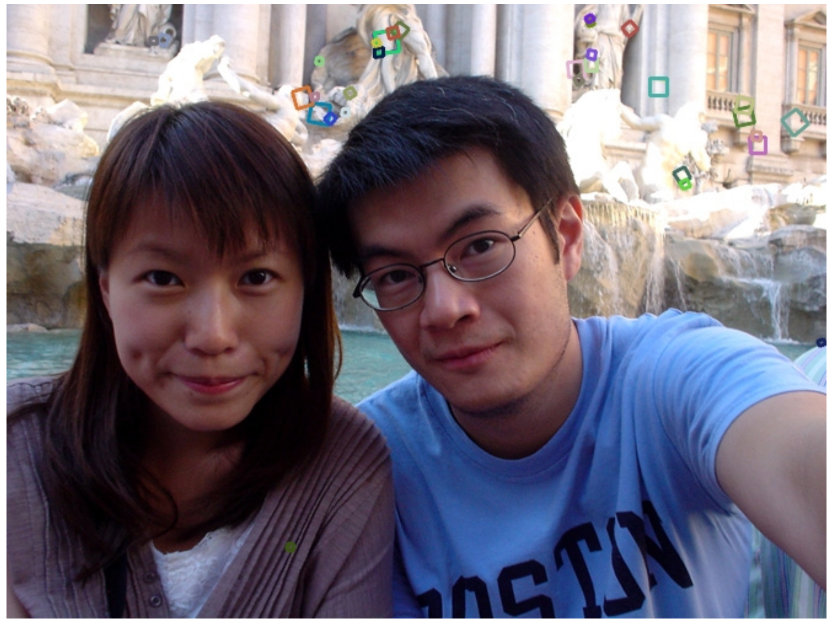

Keypoint descriptor

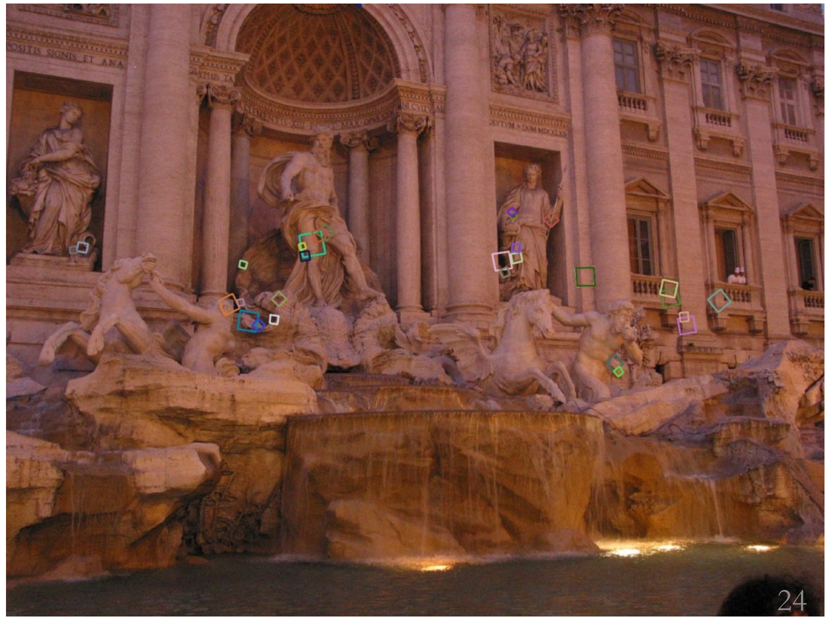
https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

Adapted from K. Grauman, D. Lowe

SIFT is highly robust

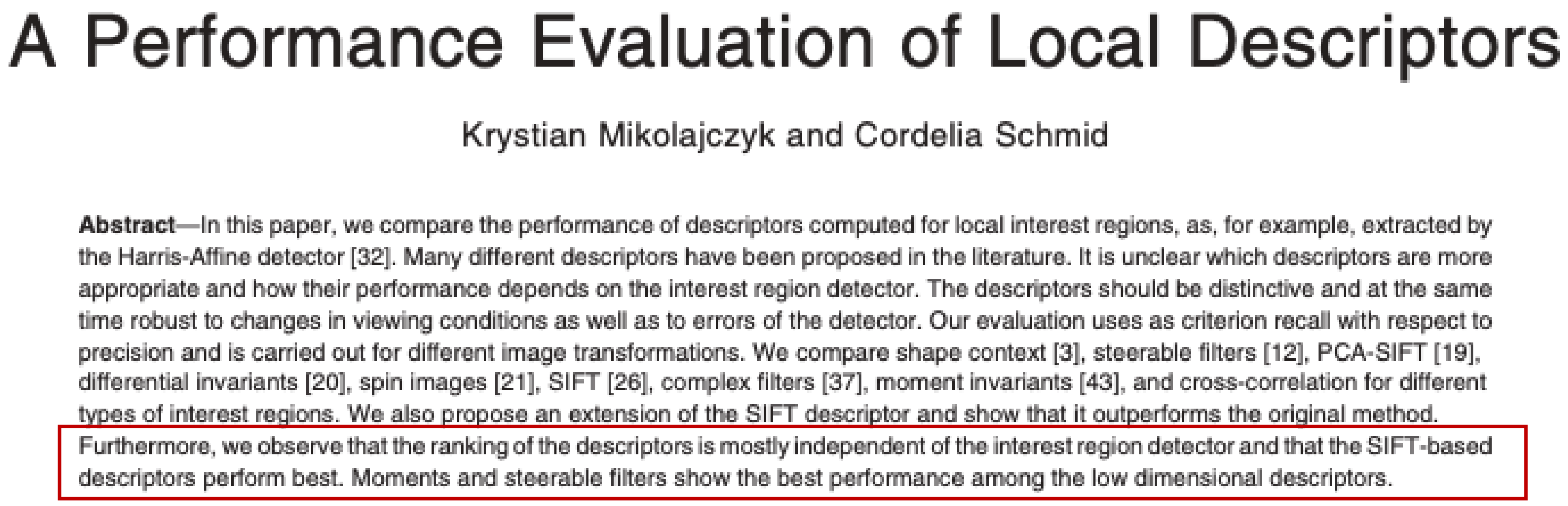
- Invariant to scale & rotation
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night
- Quick and efficient
  - SURF is an even faster follow-up derived from SIFT, using faster filters and other computational tricks
  - a nice (semi-technical) explanation [here]





Adapted from S. Seitz

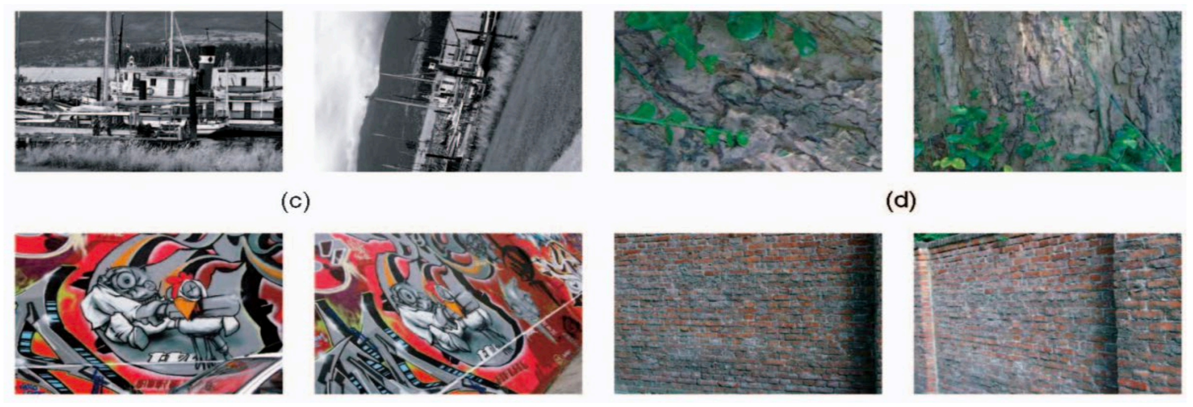
SIFT Descriptors vs. Others



A Performance Evaluation of Local Descriptors

Krystian Mikolajczyk and Cordelia Schmid

Abstract—In this paper, we compare the performance of descriptors computed for local interest regions, as, for example, extracted by the Harris-Affine detector [32]. Many different descriptors have been proposed in the literature. It is unclear which descriptors are more appropriate and how their performance depends on the interest region detector. The descriptors should be distinctive and at the same time robust to changes in viewing conditions as well as to errors of the detector. Our evaluation uses as criterion recall with respect to precision and is carried out for different image transformations. We compare shape context [3], steerable filters [12], PCA-SIFT [19], differential invariants [20], spin images [21], SIFT [26], complex filters [37], moment invariants [43], and cross-correlation for different types of interest regions. We also propose an extension of the SIFT descriptor and show that it outperforms the original method. Furthermore, we observe that the ranking of the descriptors is mostly independent of the interest region detector and that the SIFT-based descriptors perform best. Moments and steerable filters show the best performance among the low dimensional descriptors.



https://www.robots.ox.ac.uk/~vgg/research/affine/det_eval_files/mikolajczyk_pami2004.pdf

Feature Matching

Ratio of distances, ROC evaluation, Parameterization

Feature Matching

Match parts which are the same on both images

a. Find locations to match.

b. Represent surrounding region mathematically.

c. Do the matching.

Interest point **detection**.

Compute feature **descriptors**.

Feature **matching**.

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

1. Define a distance function that compares two descriptors

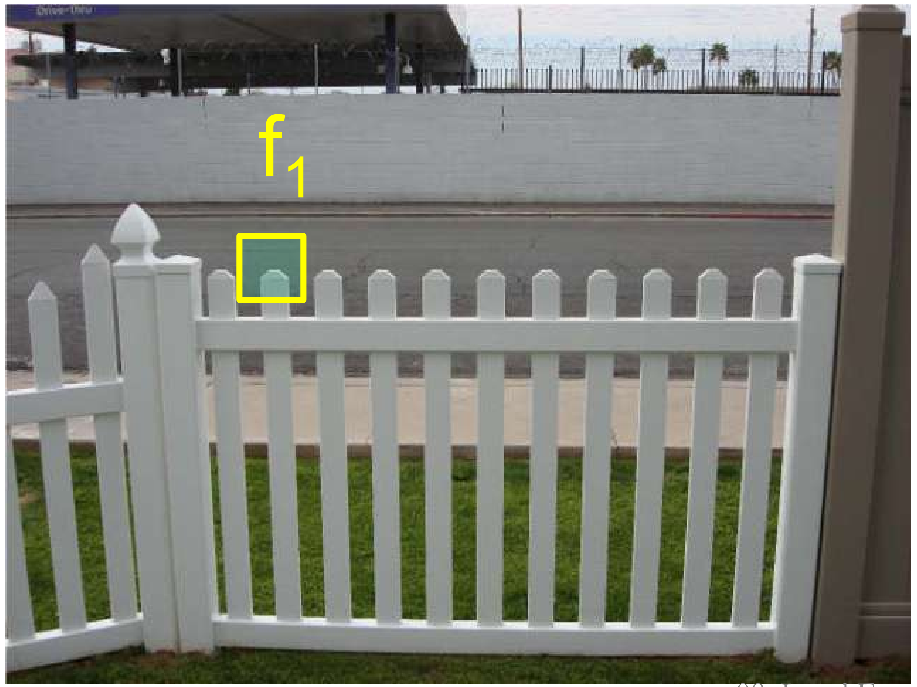
2. Test all the features in I_2, find the one with min distance

Feature Distance

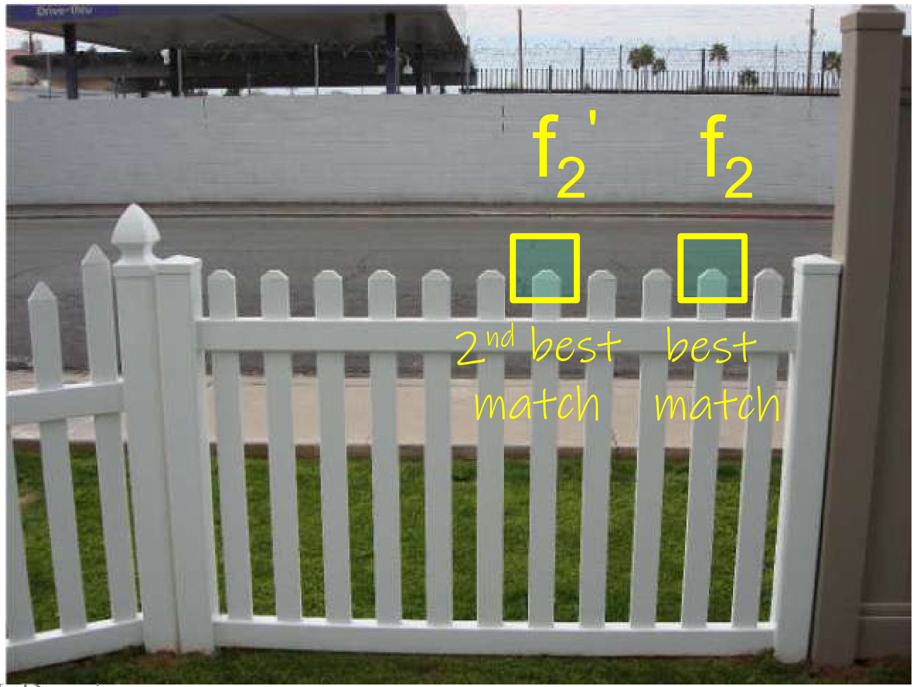
How to assess difference between two features f_1, f_2?

- Simple approach: L_2 distance, $\|f_1 - f_2\|$ (sqrt of SSD)
- Can give small distances for ambiguous (incorrect) matches
- Better approach: ratio distance = $\|f_1 - f_2\| / \|f_1 - f_2'\|$

proposed in SIFT paper



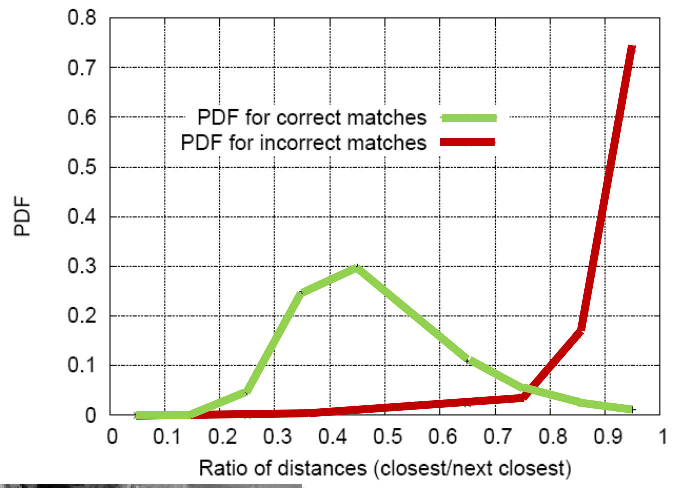
I_1



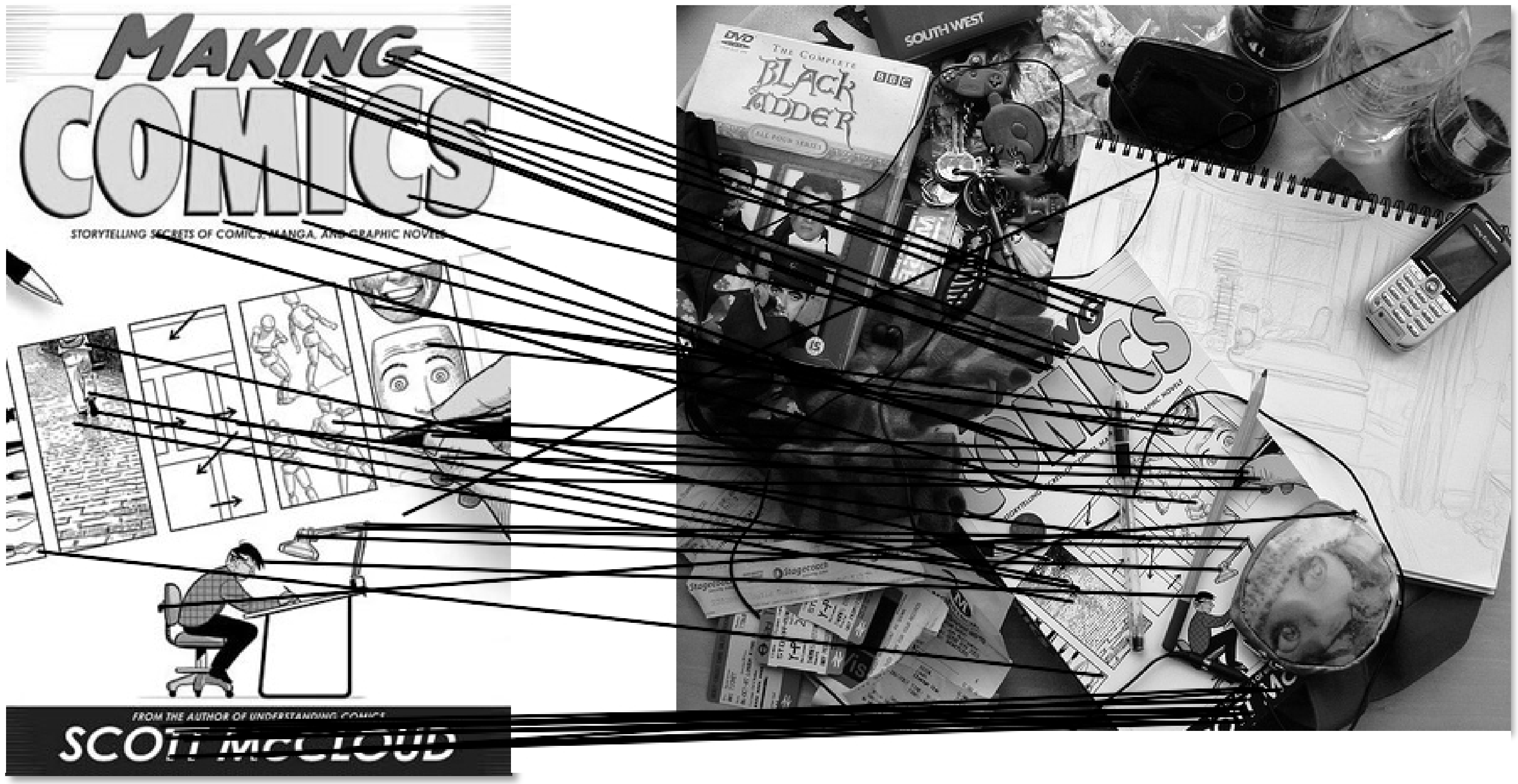
I_2

Adapted from K. Grauman

Matching SIFT Descriptors



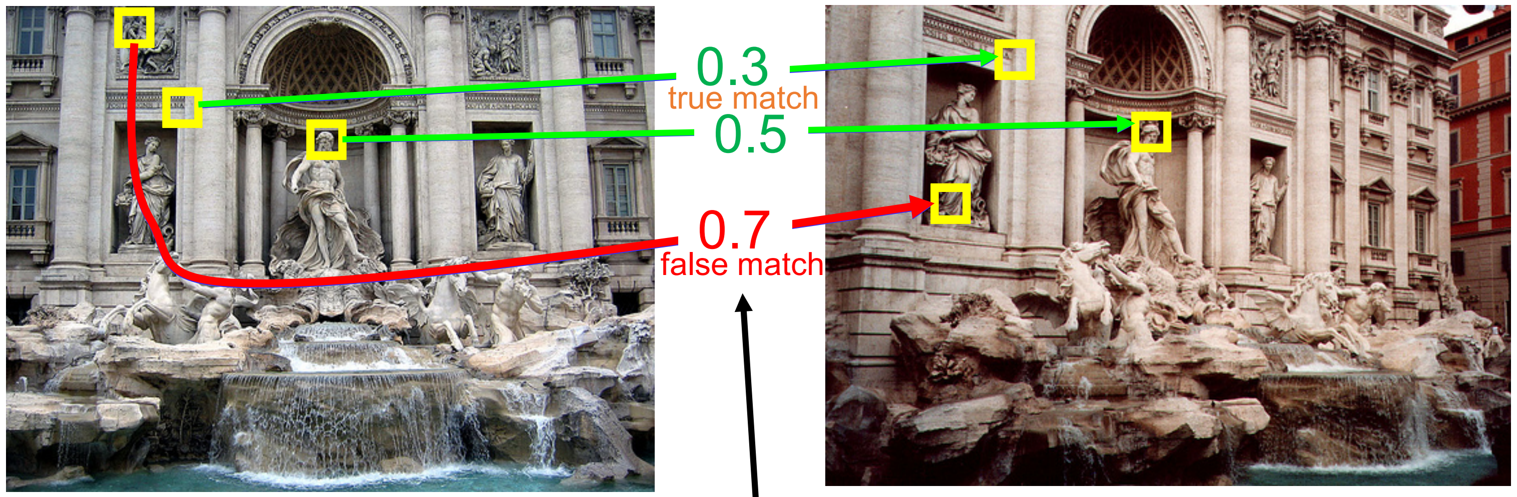
Threshold ratio of nearest to 2nd nearest descriptor



51 matches (thresholded by ratio score)

Lowe IJCV 2004

Evaluating Feature Matches



feature distance ratios

True Positives (TP): # correct matches
False Positives (FP): # incorrect matches

True Negatives (TN): keypoints which should not be matched since the corresponding keypoints were not found in the other image.

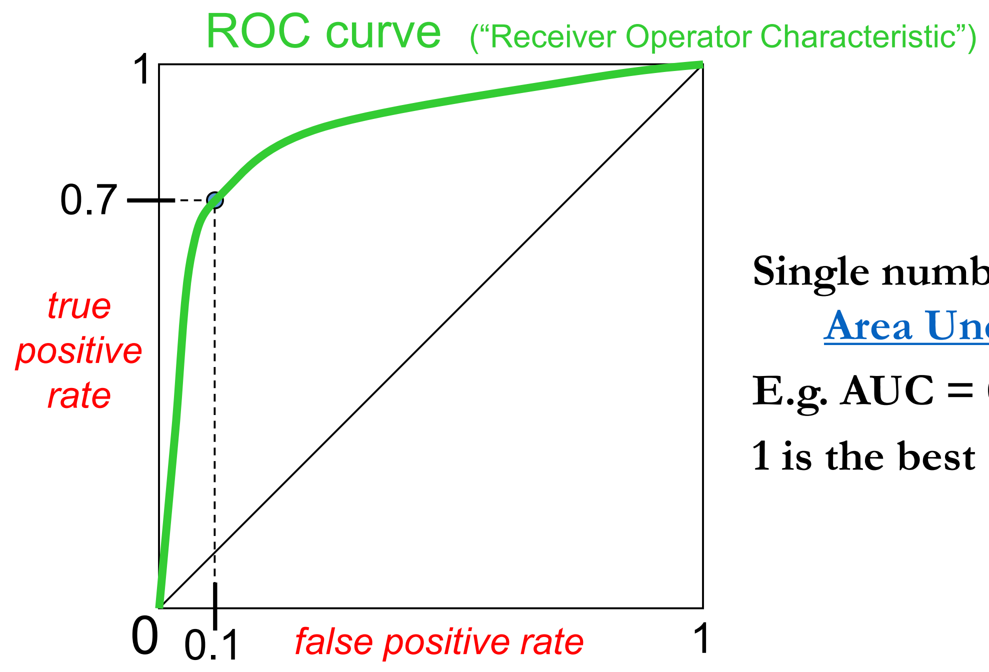
False negatives (FN): matches that we missed, i.e. corresponding keypoints which are detected in both images but not matched.

- Threshold the matches according to the distance;
- Post-thresholding, check for true vs. false positives
- The distance (ratio) threshold affects the matching performance

Adapted from K. Grauman

Evaluating Feature Matches

How can we measure the performance of a feature matcher?



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

# of “matching features” refers to total number of actual pairs of features, i.e. TPs + FNs

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

$1 - TN / (TN + FP)$
$= FP / (TN + FP)$

Single number:
Area Under the Curve (AUC)

E.g. AUC = 0.87

1 is the best

Thresholding / Precision / Recall

	Feature Matching
scoring	We are scoring wrt to feature matching across pairs of images and not feature detection (e.g. cornerness score). A lower score is a better match, since we use a distance (ratio) measure for comparison
threshold	Keep matches if below some threshold
TP	Correct match
FP	Wrong match between two features
FN	Pair of matching features not matched
TN	Detected features not part of any feature pair
Precision TP / (TP + FP)	How accurate are the feature pairs declared as matches? As the threshold goes from low to high, precision goes …
Recall TP / (TP + FN)	Was the algorithm able to find all the actual pairs of features? As threshold goes from low to high, recall goes …
Specificity TN / (TN + FP)	Can the algorithm correctly disregard the features which are not part of any pair? As threshold goes from low to high, specificity goes …

We want all of these to be high

HWQ: what happens to precision, recall and specificity as the threshold goes from low to high?

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~~

based on ratio of min vs. "next" min

Loop over all features, results in quadratic complexity wrt # of features

Use an indexing structure: e.g. search tree, hash table (insert favourite data structure and algorithm from your databases course here for efficient search / retrieval)

If we know the type of transformation from one image to another, can also help matching by directly fitting the transformation parameters.

Summary

- image descriptors mathematically characterize a region in the image
  - ideal descriptors should be robust, distinctive, compact & efficient
  - previous examples: texton histograms, mean gradients, filter bank responses
  - new examples: GIST, MOPS, SIFT
- SIFT: keypoint detector + descriptor
  - keypoint detector based on multi-scale Laplacian of Gaussians + thresholding
  - descriptor can be used stand-alone is a carefully engineered spatial histogram of local gradient orientations
- Feature matching
  - ratio of distance between best vs. next-best match to avoid ambiguous matches
  - evaluate via ROC curve, area under the curve as a single number