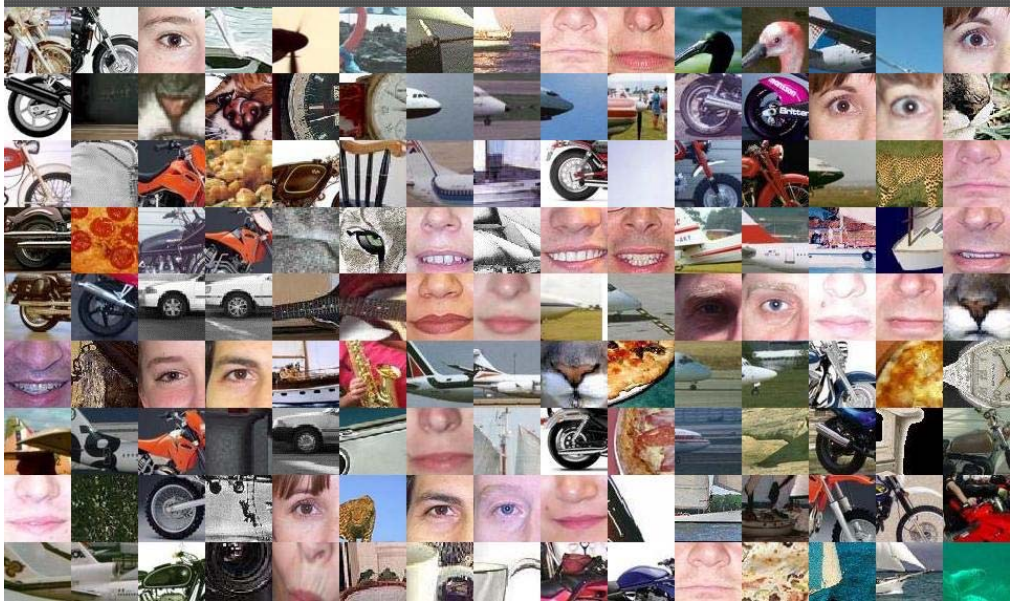
Local Descriptors

How do we describe an image patch?

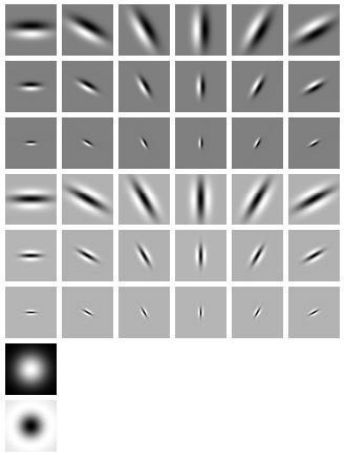
Patches with similar content should have similar descriptors.

How do we describe an image patch?



What do human use?

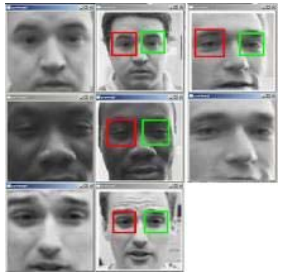
Gabor filters…

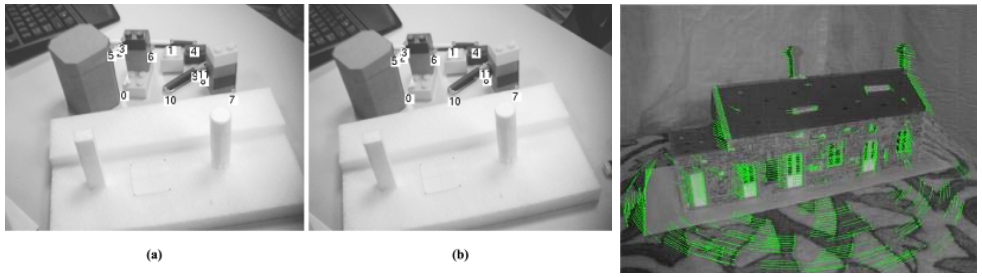


… and many other things.

Matching features

- Object Recognition
- Wide baseline matching
- Tracking/SFM

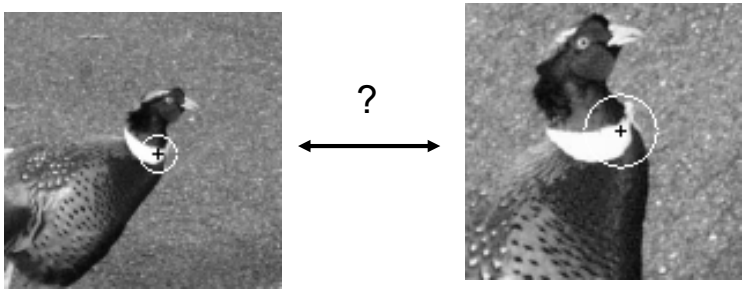




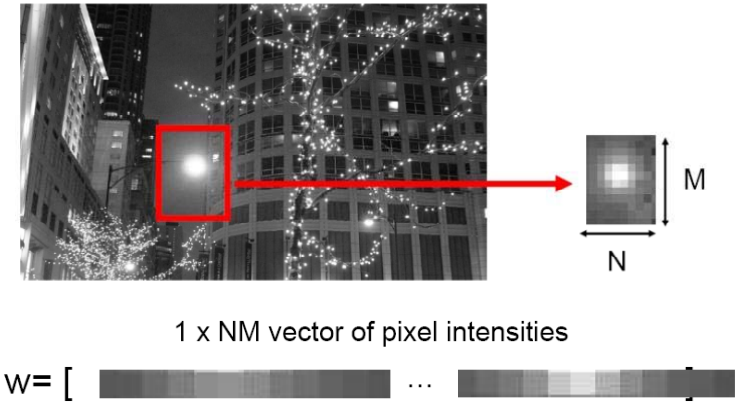
(a) (b)

Matching Application

- what information should we use for matching corresponding regions in different images?



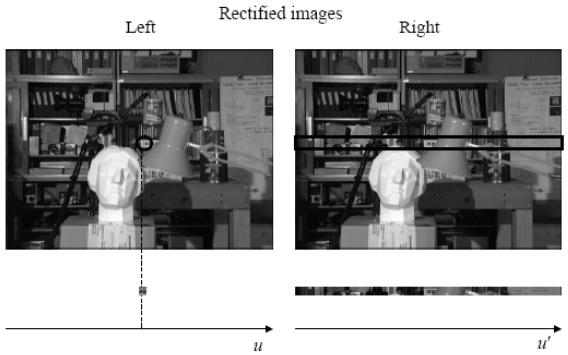
Simplest approach: correlation



- Directly compare intensities using “sum of squared differences” or “normalized cross-correlation”

Simplest approach: correlation

- Works satisfactorily when we match corresponding regions related mostly by **translation**.
  - e.g., stereo pairs, video sequence assuming small camera motion



- Hessian Matrix
  - is the square matrix of second-order partial derivatives of a function. It describes the local curvature of a function of many variables.

Given the real-valued function $f(x_1, x_2, \ldots, x_n)$,

if all second partial derivatives of *f* exist, then the Hessian matrix of *f* is the matrix

$$H(f)_{ij}(x) = D_i D_j f(x)$$

where $x = (x_1, x_2, \ldots, x_n)$ and D_i is the differentiation operator with respect to the *i*th argument

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Because f is often clear from context,

$$H(f)(x)$$

is frequently shortened to simply

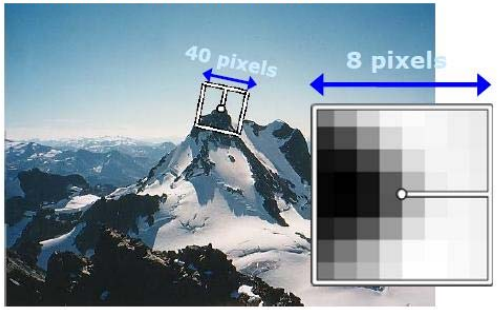
$$H(x)$$

Rotation invariance for descriptors

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

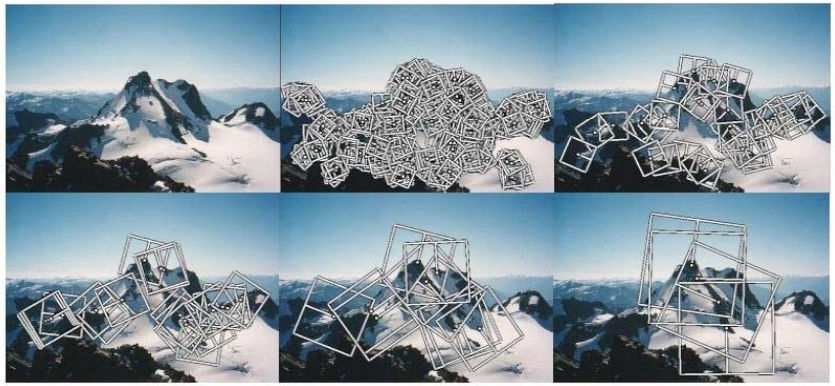
Multi-scale Oriented Patches (MOPS)

- 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.



Multi-scale Oriented Patches (MOPS)

- Extract oriented patches at multiple scales of the Gaussian pyramid.



SIFT (Scale-Invariant Feature Transform)

Lowe, D. “Distinctive image features from scale-invariant keypoints” International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

- For any object in an image, interesting points on the object can be extracted to provide a "*feature description*" of the object.
- This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects.
- To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image **scale, translation, rotation, noise and illumination.**
- Such points usually lie on high-contrast regions of the image, such as object edges.

Simplest approach: correlation

- Sensitive to small variations with respect to:
  - Location
  - Pose
  - Scale
  - Intra-class variability
- Poorly distinctive!
- We will discuss a powerful descriptor called SIFT

Types of invariance

- Illumination

Types of invariance

- Illumination
- Scale

Types of invariance

- Illumination
- Scale
- Rotation
- Affine

Types of invariance

- Illumination
- Scale
- Rotation

Types of invariance

- Illumination
- Scale
- Rotation
- Affine
- Full Perspective

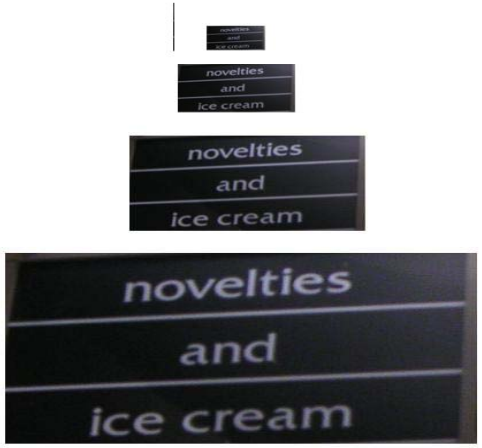
How to achieve scale invariance

- Pyramids
  - Divide width and height by 2
  - Take average of 4 pixels for each pixel (or Gaussian blur)
  - Repeat until image is tiny
  - Run filter over each size image and hope its robust
- Scale Space (DOG method)

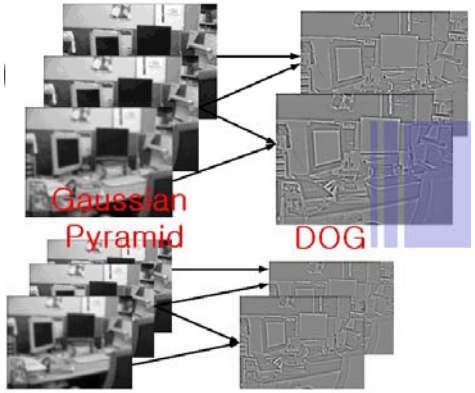
How to achieve scale invariance

- Pyramids
- Scale Space (DOG method)
  - Pyramid but fill gaps with blurred images
  - Like having a nice linear scaling without the expense
  - Take features from differences of these images
  - If the feature is repeatably present in between Difference of Gaussians it is Scale Invariant and we should keep it.

Pyramids



Differences Of Gaussians



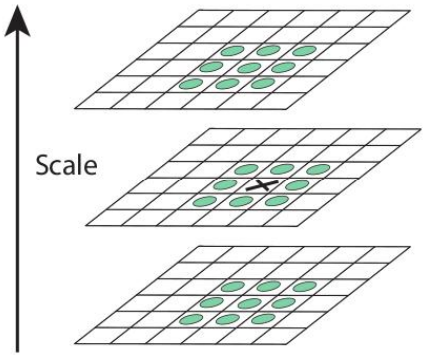
SIFT (Scale-Invariant Feature Transform)

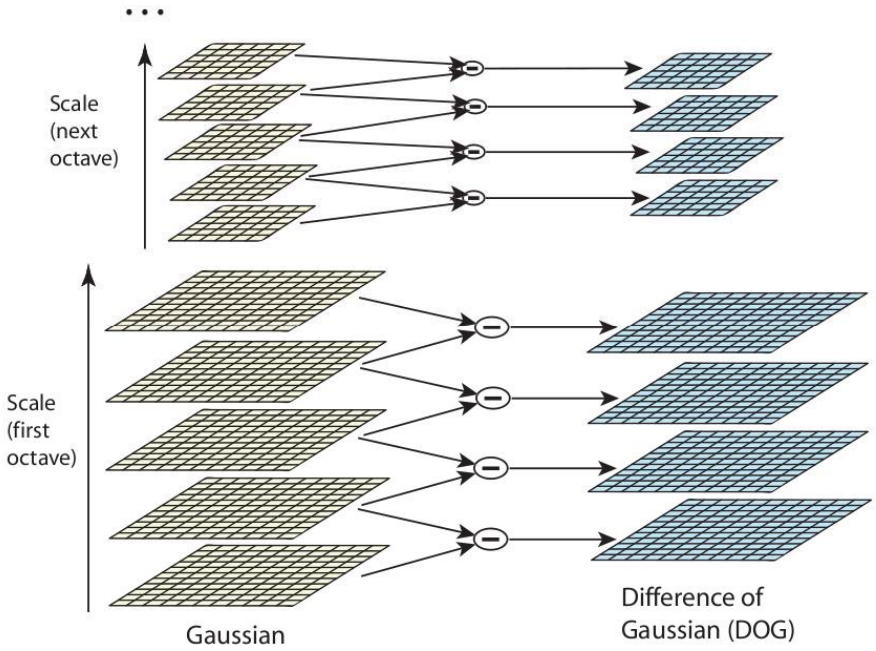
Step 1. Scale-space extrema detection

- This is the stage where the interest points, which are called keypoints in the SIFT framework, are detected.
- For this, the image is convolved with Gaussian filters at different scales, and then the difference of successive Gaussian-blurred images are taken.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma)$$

- Keypoints are then taken as maxima/minima of the DoG that occur at multiple scales.
- This is done by comparing each pixel in the DoG images to its eight neighbors at the same scale and nine corresponding neighboring pixels in each of the neighboring scales. If the pixel value is the maximum or minimum among all 26 compared pixels, it is selected as a candidate keypoint.
- Note each keypoint has an associate scale





SIFT (Scale-Invariant Feature Transform)

- **Step 2. Keypoint filtering**
- **Discarding low-contrast keypoints**
- **Eliminating edge responses**

•The DoG function will have strong responses along edges. Most of this selected keypoints have poorly determined locations along the edge.

•We can discard them by solving for the eigenvalues of the Hessian Matrix:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

•The eigenvalues of **H** are proportional to the principal curvatures of D. It turns out that the ratio of the two eigenvalues (α,β) with $\alpha > \beta$)

$r = \alpha/\beta$ is sufficient for SIFT's purposes.

SIFT

•The ratio: $\mathrm{R} = \mathrm{Tr}(\mathbf{H})^2 / \mathrm{Det}(\mathbf{H})$

•can be shown to be equal to: $(r+1)^2/r$

•which depends only on the ratio of the eigenvalues rather than their individual values.

•R is minimum (R=4) when the eigenvalues are equal to each other.

•The higher the absolute difference between the two eigenvalues, the higher the value of R.

•A threshold on eigenvalue ratio is used to reject poorly localized keypoints. Values from 10 to 12 are usually used.

SIFT

•An orientation histogram with 36 bins is formed, with each bin covering 10 degrees.

•Each sample in the neighboring window added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint.

•The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint.

•In the case of multiple orientations being assigned, an additional keypoint is created having the same location and scale as the original keypoint for each additional orientation.

SIFT (Scale-Invariant Feature Transform)

- **Step 3. Orientation assignment**
- This is the key step in achieving invariance to rotation

•For every keypoint its gradient magnitude and orientation are precomputed using pixel differences (to be used as a reference):

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

•The magnitude and direction calculations for the gradient are done for every pixel in a neighboring region around the keypoint in the Gaussian-blurred image L.

Actual SIFT stage output

Keypoint detection

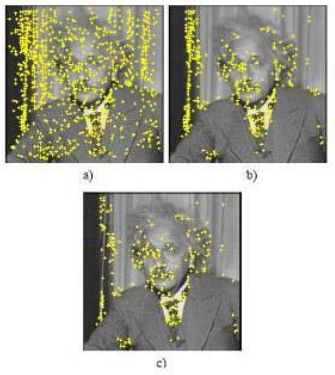


Figure 5: a) Maxima of DoG across scales. b) Remaining keypoints after removal of low contrast points. C) Remaining keypoints after removal of edge responses (bottom).

Final keypoints with selected orientation and scale

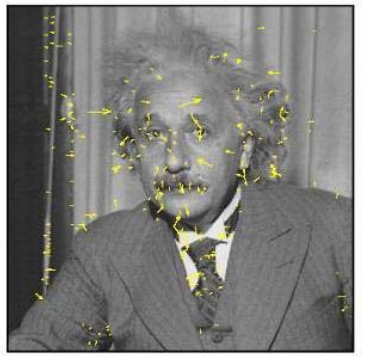


Figure 6: Extracted keypoints, arrows indicate scale and orientation.

SIFT (Scale-Invariant Feature Transform)

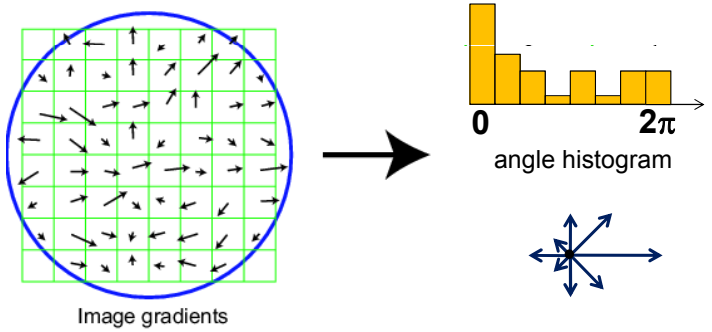
- **Step 4. Keypoint descriptor**
- we want to compute a descriptor vector for each keypoint such that it is highly distinctive and partially invariant to the remaining variations such as illumination, 3D viewpoint, etc.

•The descriptor consist of a set of 16 orientation histograms with 8 bins each (128 elements).

•These histograms are computed from magnitude and orientation values of samples in a 16 x 16 (64 pixel) region around the keypoint such that each histogram contains samples from a 4 x 4 pixel subregion of the original neighborhood region.

Basic idea:

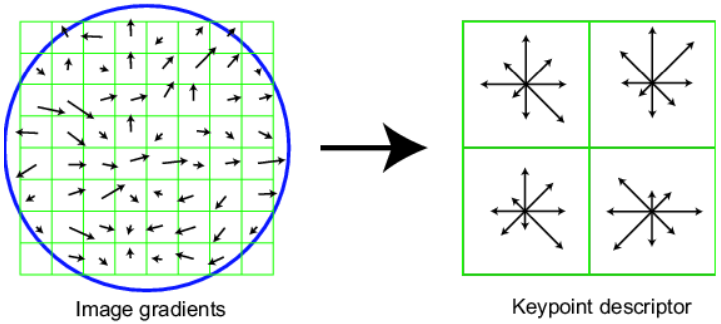
- Take 16x16 square window around detected feature
- Compute gradient orientation for each pixel
- Create histogram over edge orientations weighted by magnitude



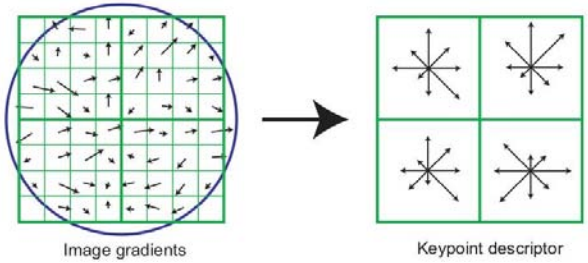
Adapted from slide by David Lowe

Full version

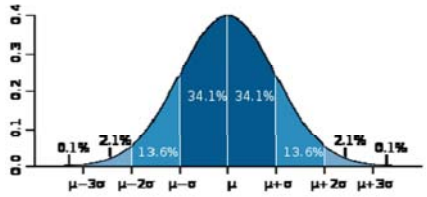
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = **128 dimensional descriptor**



Adapted from slide by David Lowe



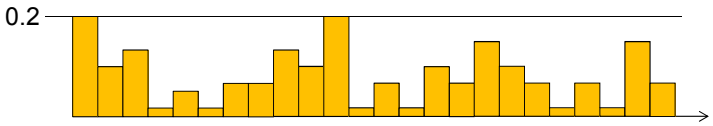
•The magnitudes are further weighted by a Gaussian function with σ equal to ½ width of the descriptor window.

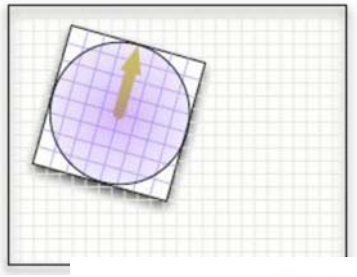


Full version

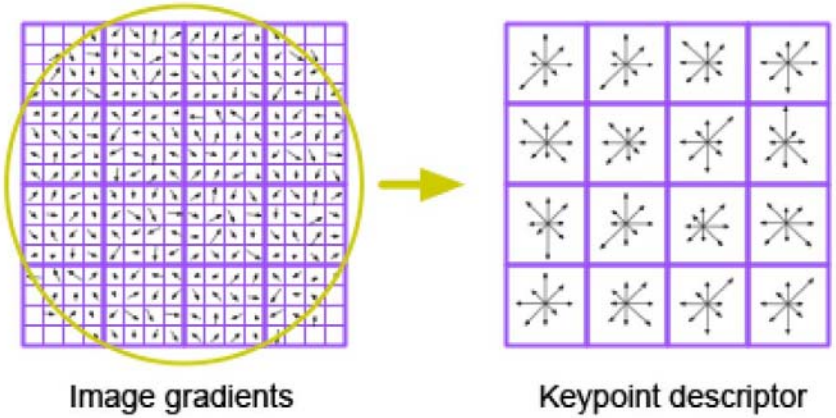
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor
- This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination.
- To reduce the effects of non-linear illumination a threshold of 0.2 is applied and the vector is again normalized.

$$\sum_i d_i^2 = 1 \quad \text{such that:} \quad d_i < 0.2$$





Remember STEP 3: Rotation invariance



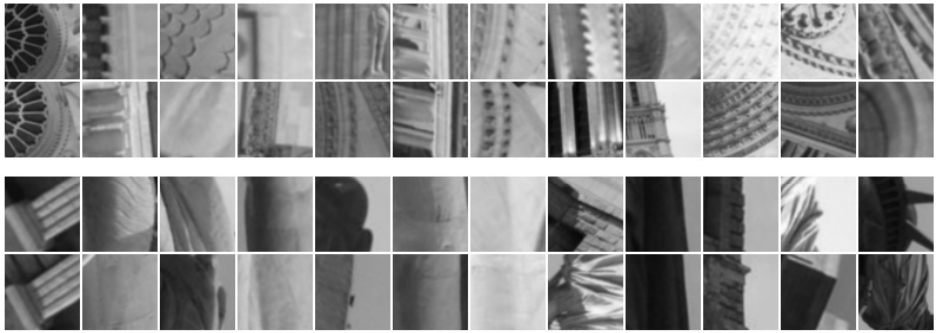
Properties of SIFT

Extraordinarily robust matching technique

- Can handle changes in viewpoint
  - Up to about 30 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - http://people.csail.mit.edu/albert/ladypack/wiki/index.php/Known_implementations_of_SIFT

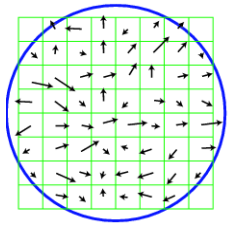
When does SIFT fail?

Patches SIFT thought were the same but aren't:



PCA-SIFT

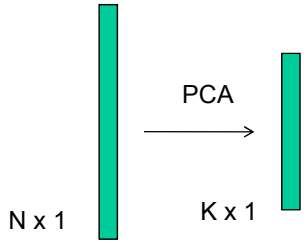
- Steps 1-3 are the same; Step 4 is modified.
- Take a 41 x 41 patch at the given scale, centered at the keypoint, and normalized to a canonical direction.



Yan Ke and Rahul Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors", **Computer Vision and Pattern Recognition**, 2004

PCA-SIFT

- Reduce the dimensionality of the vector using Principal Component Analysis (PCA)
  - e.g., from 3042 to 36

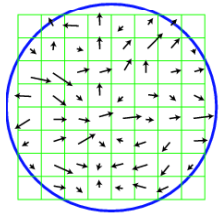


$$A_{KxN} I_{Nx1} = I'_{Kx1}$$

- Proved to be less discriminatory than SIFT.

PCA-SIFT

- Instead of using weighted histograms, concatenate the horizontal and vertical gradients into a long vector.
- Normalize vector to unit length.



2 x 39 x 39 = 3042 vector

Yan Ke and Rahul Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors", **Computer Vision and Pattern Recognition**, 2004

SURF: Speeded Up Robust Features

- Fast implementation of a variation of the SIFT descriptor.
- Key idea: fast approximation of (i) Hessian matrix and (ii) descriptor using "integral images".

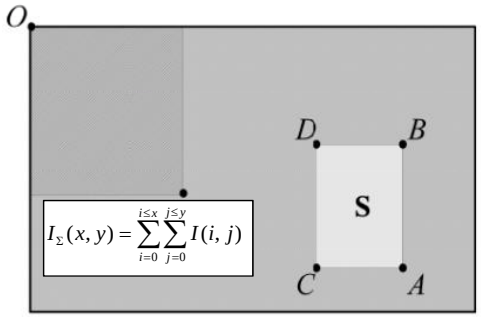
$$\mathcal{H}(\mathrm{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathrm{x}, \sigma) & L_{xy}(\mathrm{x}, \sigma) \\ L_{xy}(\mathrm{x}, \sigma) & L_{yy}(\mathrm{x}, \sigma) \end{bmatrix},$$

- What is an "integral image"?

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, "SURF: Speeded Up Robust Features", **European Computer Vision Conference (ECCV),** 2006.

Integral Image

- The integral image $I_{\Sigma}(x,y)$ of an image I(x, y) represents the sum of all pixels in I(x,y) of a rectangular region formed by (0,0) and (x,y).



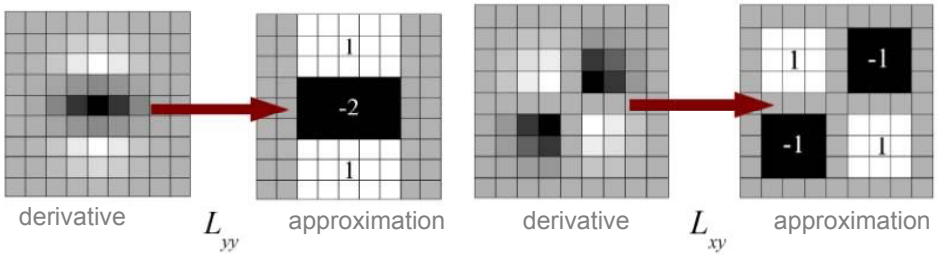
Using integral images, it takes only four array references to calculate the sum of pixels over a rectangular region of any size.

$$\mathbf{S}=A-B-C+D$$

SURF: Speeded Up Robust Features

- Approximate L_{xx}, L_{yy}, and L_{xy} using box filters.

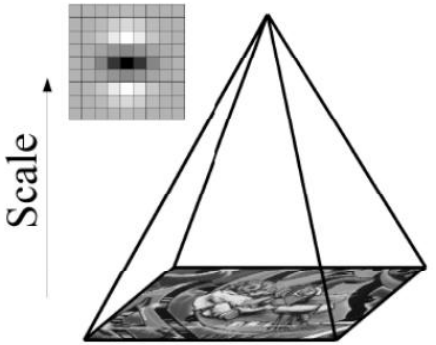
(box filters shown are 9 x 9 – good approximations for a Gaussian with σ=1.2)



- Can be computed very fast using integral images!

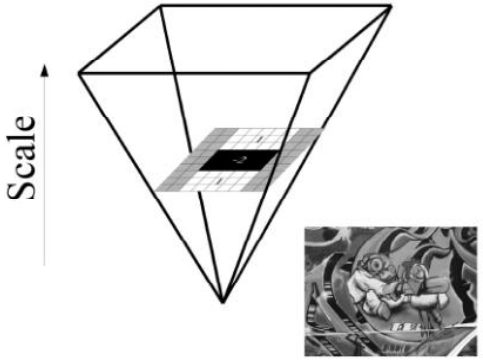
SURF: Speeded Up Robust Features (cont'd)

- In SIFT, images are repeatedly smoothed with a Gaussian and subsequently sub-sampled in order to achieve a higher level of the pyramid.



SURF: Speeded Up Robust Features (cont'd)

- Alternatively, we can use filters of larger size on the original image.
- Due to using integral images, filters of any size can be applied at exactly the same speed!



(see paper for details on how the filter size and octaves are computed)

SURF: Speeded Up Robust Features (cont'd)

- Approximation of H:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix},$$

Using DoG

$$SIFT: \; H_{approx}^{SIFT} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

$$SURF: \; H_{approx}^{SURF} = \begin{bmatrix} \hat{L}_{xx} & \hat{L}_{xy} \\ \hat{L}_{yx} & \hat{L}_{yy} \end{bmatrix}$$

Using box filters

SURF: Speeded Up Robust Features

- Once interest points have been localized both in space and scale, the next steps are:
  - (1) Orientation assignment
  - (2) Keypoint descriptor

SURF: Speeded Up Robust Features (cont'd)

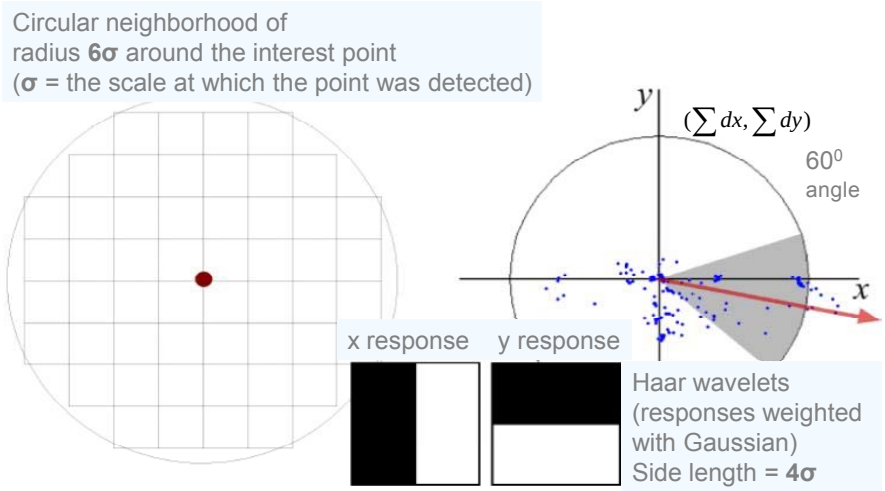
- Instead of using a different measure for selecting the location/scale and singularity of interest points (like in SIFT), SURF uses the determinant of H_{approx}^{SURF} to find both.

- Determinant elements must be weighted to obtain a good approximation of det(H):

$$\det(H_{approx}^{SURF}) = \hat{L}_{xx}\hat{L}_{yy} - (0.9\hat{L}_{xy})^2$$

SURF: Speeded Up Robust Features

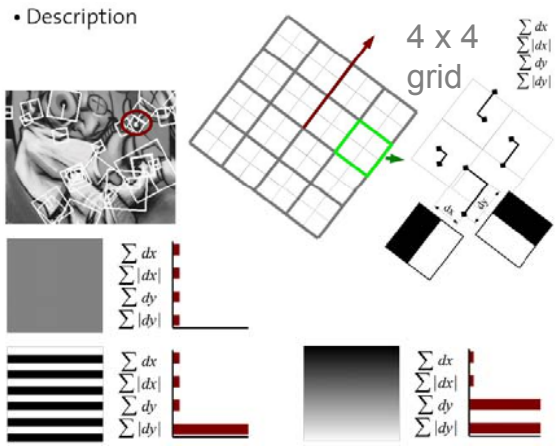
- Orientation assignment



Can be computed very fast using integral images!

SURF: Speeded Up Robust Features

- Keypoint descriptor (square region of size 20σ)



- Sum the response over each sub-region for d_x and d_y separately.
- To bring in information about the polarity of the intensity changes, extract the sum of absolute value of the responses too.

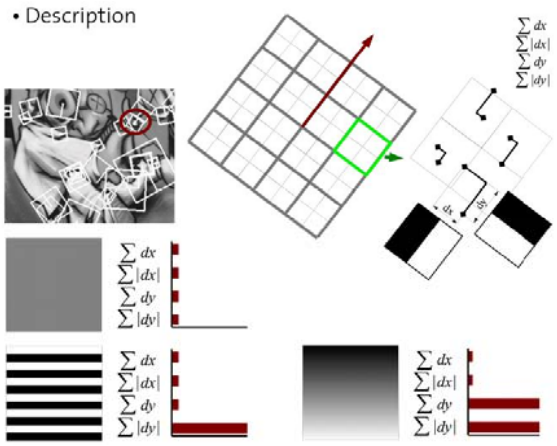
Feature vector size:
4 x 16 = 64

SURF: Speeded Up Robust Features

- Has been reported to be 3 times faster than SIFT.
- Less robust to illumination and viewpoint changes compared to SIFT.

K. Mikolajczyk and C. Schmid,"A Performance Evaluation of Local Descriptors", **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 27, no. 10, pp. 1615-1630, 2005.

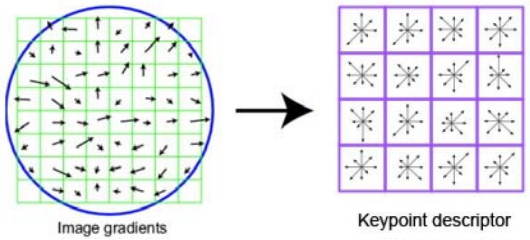
SURF: Speeded Up Robust Features



- SURF-128
  - The sum of d_x and $|d_x|$ are computed separately for points where $d_y < 0$ and $d_y > 0$
  - Similarly for the sum of d_y and $|d_y|$
  - **More discriminatory!**

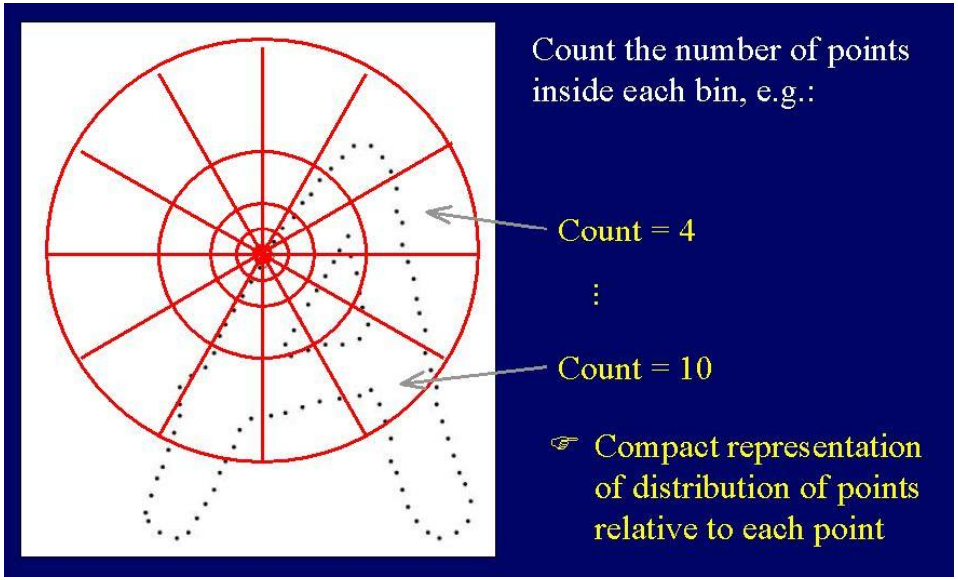
Gradient location-orientation histogram (GLOH)

- Compute SIFT using a log-polar location grid:
  - 3 bins in radial direction (i.e., radius 6, 11, and 15)
  - 8 bins in angular direction
- Gradient orientation quantized in 16 bins.
- Total: (2x8+1)*16=272 bins →PCA.



K. Mikolajczyk and C. Schmid,"A Performance Evaluation of Local Descriptors", **IEEE Trans. on Pattern Analysis and Machine Intelligence**, vol. 27, no. 10, pp. 1615-1630, 2005.

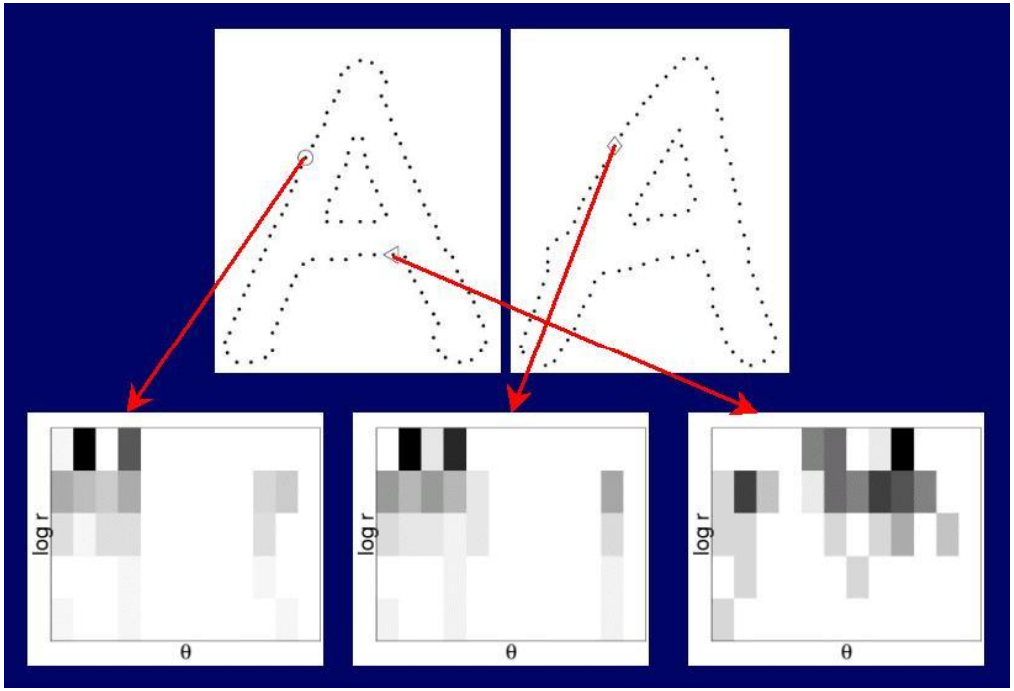
Shape Context



Shape Context

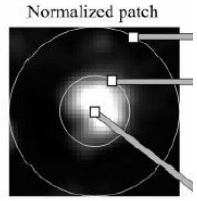
- Extension to a 3D histogram of edge point locations **and orientations**.
  - Edges are extracted by the Canny edge detector.
  - Location is quantized into 9 bins (using a log-polar coordinate system).
  - Orientation is quantized in 4 bins (i.e., (horizontal, vertical, and two diagonals).
- Total number of features: 4 x 9 = 36

K. Mikolajczyk and C. Schmid,"A Performance Evaluation of Local Descriptors", **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 27, no. 10 pp. 1615-1630, 2005.



Spin image

- A histogram of quantized pixel locations and intensity values.
  - A normalized histogram is computed for each of five rings centered on the region.
  - The intensity of a normalized patch is quantized into 10 bins.
- Total number of features: 5 x 10 = 50

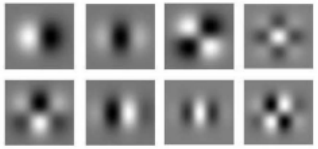


K. Mikolajczyk and C. Schmid,"A Performance Evaluation of Local Descriptors", **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 27, no. 10, pp. 1615-1630, 2005.

Differential Invariants

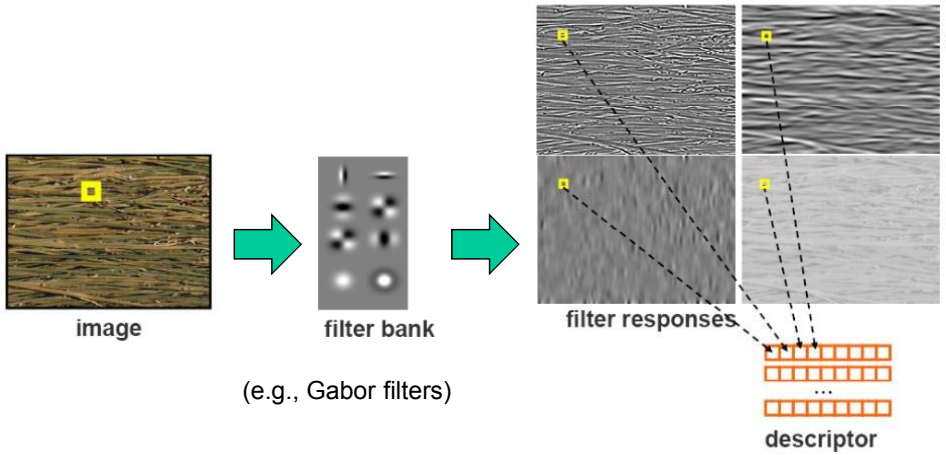
- "Local jets" of derivatives obtained by convolving the image with Gaussian derivates.
- Derivates are computed at different orientations by rotating the image patches.

Example: some Gaussian derivatives up to fourth order (dimension=14)



$$\begin{pmatrix} I(x,y)*G(\sigma) \\ I(x,y)*G_x(\sigma) \\ I(x,y)*G_y(\sigma) \\ I(x,y)*G_{xx}(\sigma) \\ I(x,y)*G_{xy}(\sigma) \\ I(x,y)*G_{yy}(\sigma) \\ \vdots \end{pmatrix} \Rightarrow \text{invariants}$$

Bank of Filters



(e.g., Gabor filters)

Moment Invariants

- Moments are computed for derivatives of an image patch using:

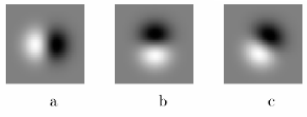
$$M_{pq}^{a} = \frac{1}{xy}\sum_{x,y} x^p y^q [I_d(x,y)]^a$$

where p+q is the order, a is the degree, and I_d is the image gradient in direction d.

- Derivatives are computed in x and y directions.
- Typical Dimension = 2*10 (without M_{00})

Bank of Filters: Steerable Filters

$$\begin{aligned} R_1^{0°} &= G_1^0 * I \\ R_1^{90°} &= G_1^{90} * I \\ \text{then} \\ R_1^{\theta} &= \cos(\theta) R_1^{0°} + \sin(\theta) R_1^{90°} \end{aligned}$$



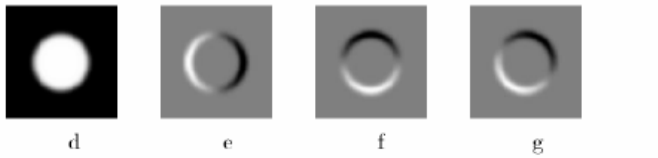
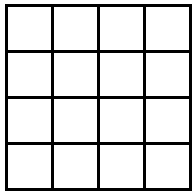


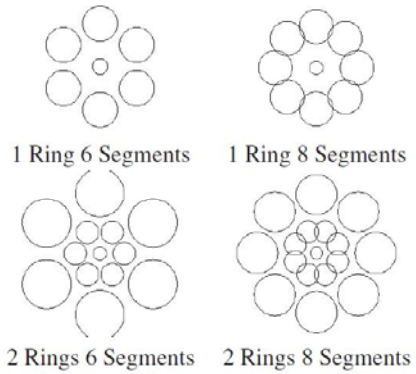
Figure 1: Example of steerable filters. (a) $G_1^{0°}$, first derivative with respect to x (horizontal) of a Gaussian. (b) $G_1^{90°}$, which is $G_1^{0°}$, rotated by 90°. From a linear combination of these two filters, one can create G_1^{θ}, an arbitrary rotation of the first derivative of a Gaussian. (c) $G_1^{90°}$, formed by $\frac{1}{2}G_1^{0°} + \frac{\sqrt{3}}{2}G_1^{90°}$. The same linear combinations used to synthesize G_1^{θ} from the basis filters will also synthesize the response of an image to G_1^{θ} from the responses of the image to the basis filters: (d) Image of circular disk. (e) $G_1^{0°}$ (at a smaller scale than pictured above) convolved with the disk, (d). (f) $G_1^{90°}$ convolved with (d). (g) $G_1^{30°}$ convolved with (d), obtained from $\frac{1}{2}$ [image e] $+\frac{\sqrt{3}}{2}$ [image f].

Other methods: Daisy



SIFT

Circular gradient binning



Daisy

Picking the best DAISY, S. Winder, G. Hua, M. Brown, CVPR 09

Other methods: BiCE

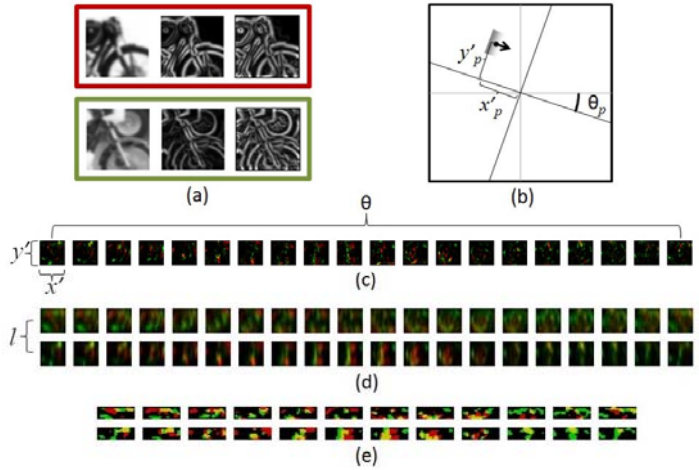
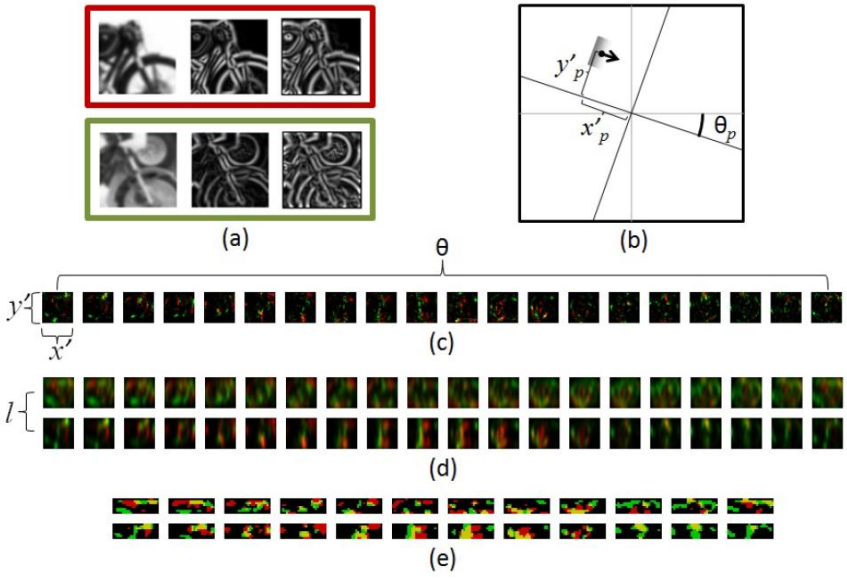


Fig. 1. Processing pipeline: (a) Two matching patches (left to right): Original patch, gradients g_p and normalized gradients $\hat{g}_p$, (b) an illustration of the coordinate frame used for the orientation dependent binning, (c) the edge histogram with the top patch shown in red and bottom patch shown in green (yellow denotes agreement), (d) the histogram after blurring and spliting the edges into two sets of bins based on the edge's local linear length, (e) final binarized descriptor, red is the top patch, green is the bottom patch and yellow denotes agreement.

Other methods: BiCE



Binary Coherent Edge Descriptors C. L. Zitnick , *ECCV* 2010.

Other methods: BRIEF

Randomly sample pair of pixels a and b. Set 1 if a > b, else 0.
Store binary vector. Compare with XOR (Hamming Distance)

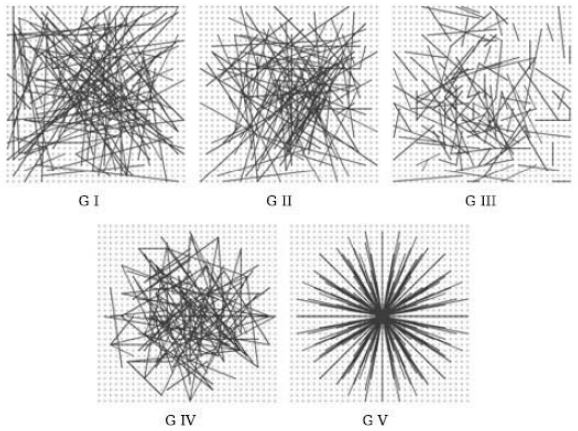
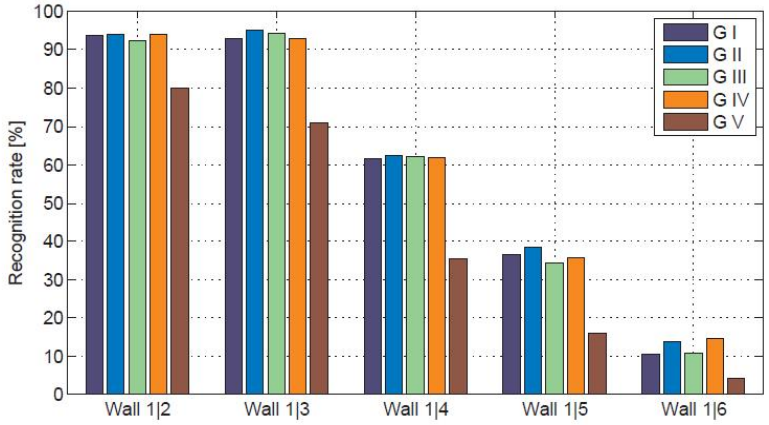


Fig. 2. Different approaches to choosing the test locations. All except the righmost one are selected by random sampling. Showing 128 tests in every image.

Other methods: BRIEF



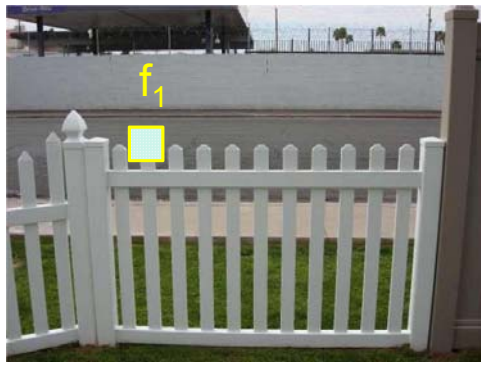
Recognition rate for the five different test geometries introduced

BRIEF: binary robust independent elementary features, Calonder, V Lepetit, C Strecha, ECCV 2010

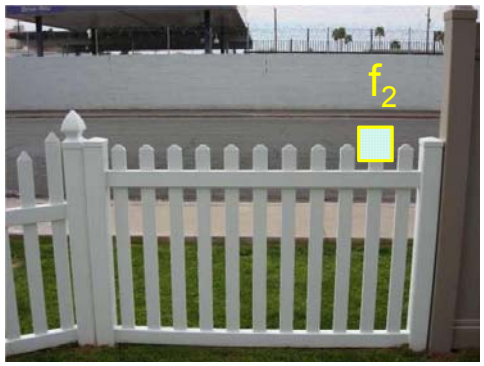
Feature distance

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

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches



I_1



I_2

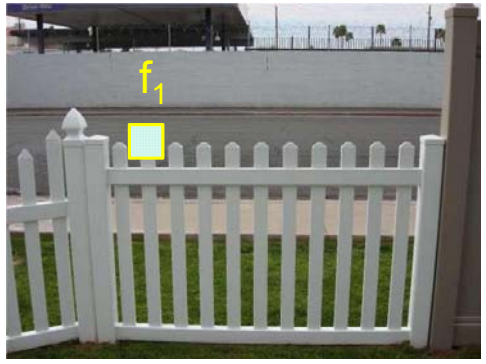
Feature distance

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

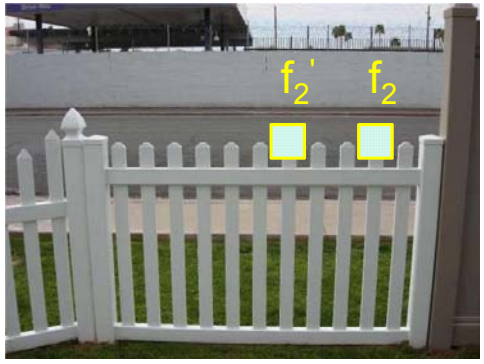
Feature distance

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 large values (~1) for ambiguous matches

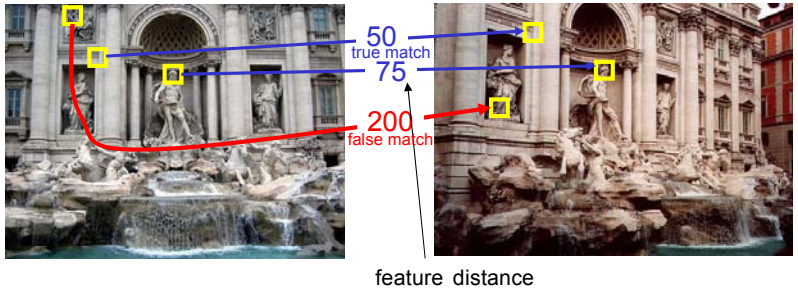


I_1



I_2

Eliminating bad matches

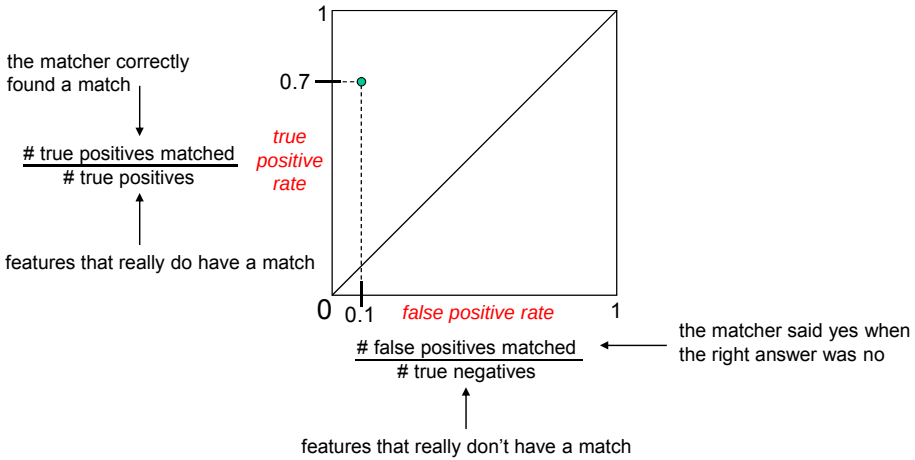


Throw out features with distance > threshold

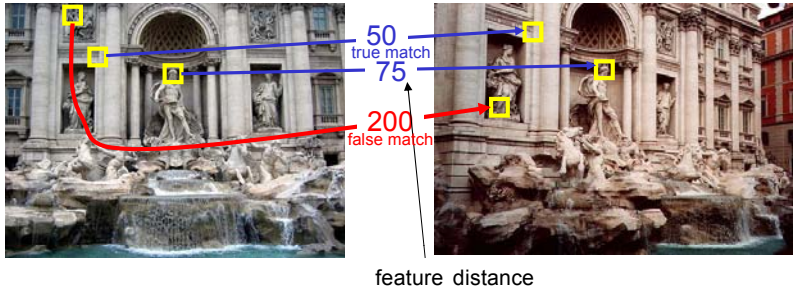
- How to choose the threshold?

Evaluating the results

How can we measure the performance of a feature matcher?



True/false positives

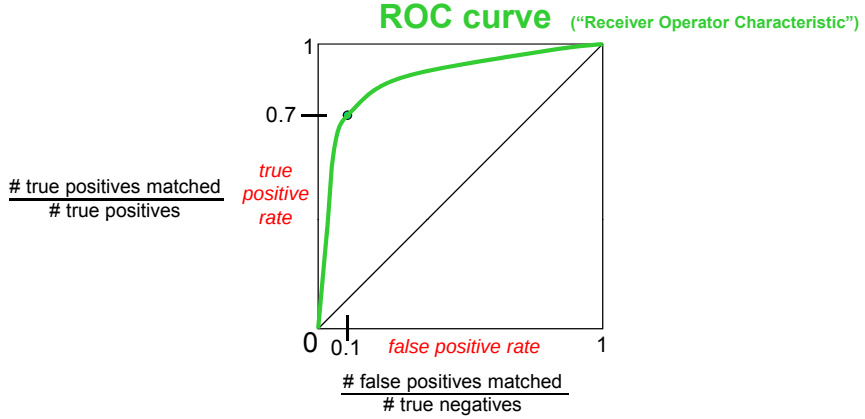


The distance threshold affects performance

- 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

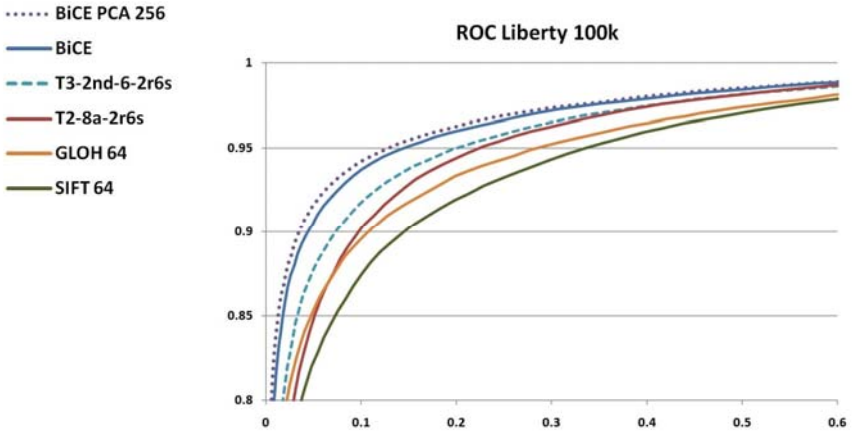
How can we measure the performance of a feature matcher?



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

Some actual ROC curves



Binary Coherent Edge Descriptors C. L. Zitnick , *ECCV* 2010.

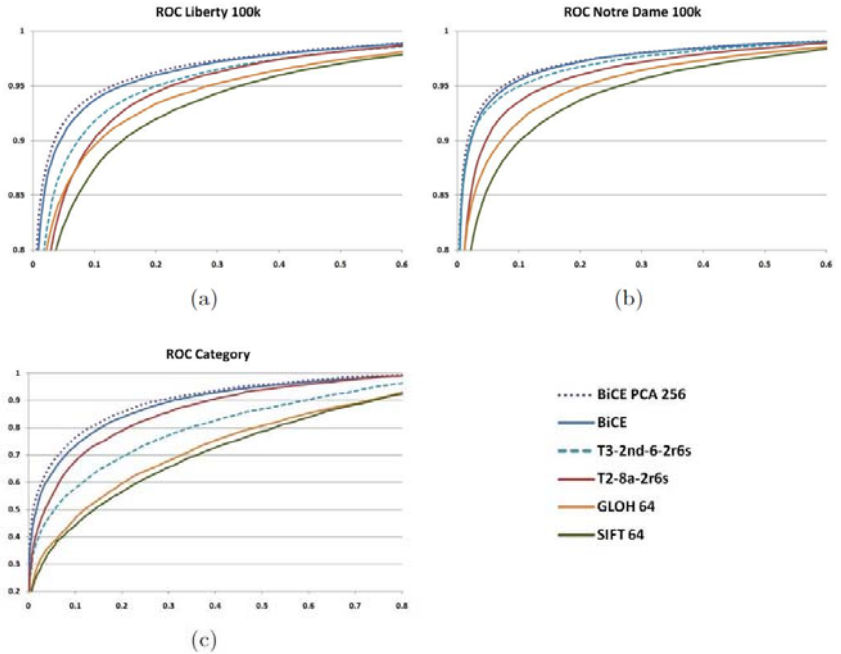


Fig. 3. ROC curves for (a) Liberty, (b) Notre Dame and (c) Category datasets. Notice the plotted ranges vary from (a,b) to the more difficult dataset of (c).