BMB5113
COMPUTER VISION

DESCRIPTORS

Local Features and Alignment

- To match or align images
  - Global methods sensitive to occlusion, lighting, parallax effects.
  - So look for local features that match well.
  - How would you do it by eye?

Local Features and Alignment

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

[Darya Frolova and Denis Simakov]

Main Questions

- Where will the interest points come from?
  - What are salient features that we'll *detect* in multiple views?
- How to *describe* a local region?
- How to establish *correspondences*, i.e., compute matches?

Local Features and Alignment

- Problem 1:
  - Detect the *same* point *independently* in both images

no chance to match!

We need a **repeatable** detector

Local Features and Alignment

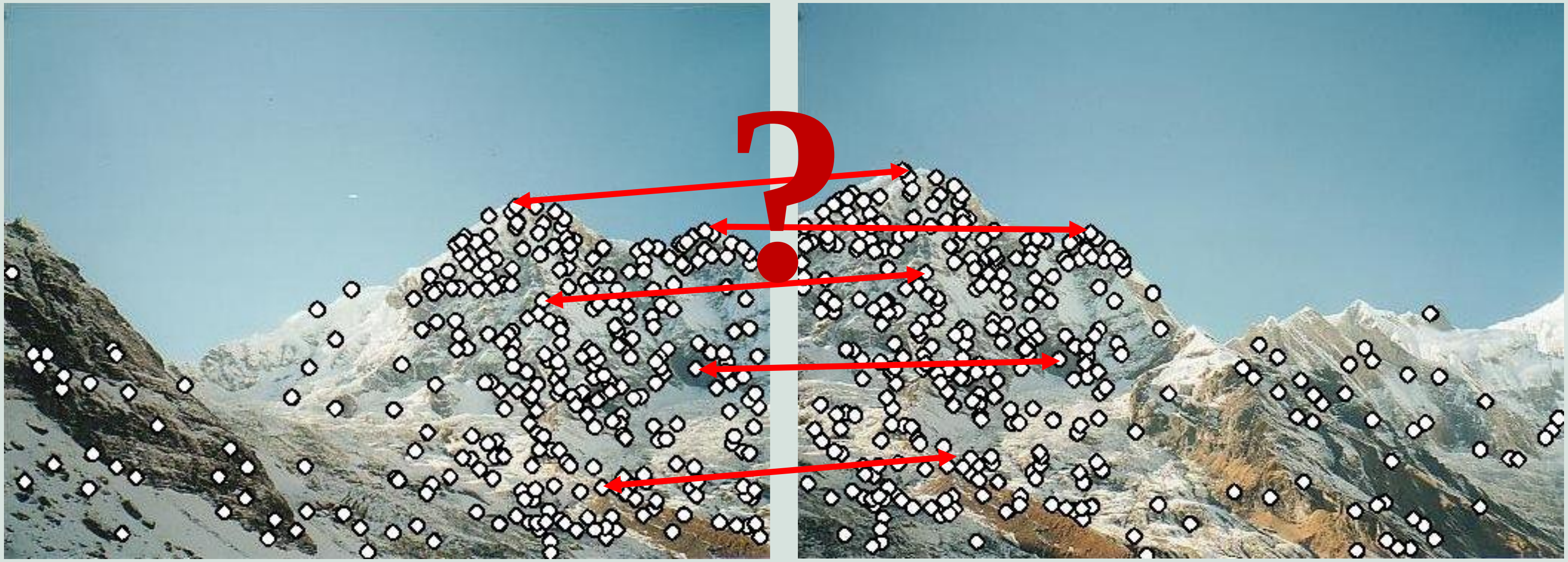
- Problem 2:
  - For each point correctly recognize the corresponding one

We need a reliable and **distinctive** descriptor

Local Descriptors

- There are a lot of methods to detect points.

How to *describe* them for matching?



Point descriptor should be:

1. Invariant
2. Distinctive

Local Descriptors

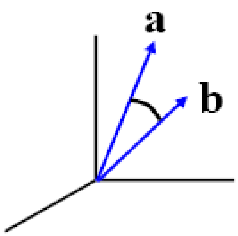
- Simplest descriptor: list of intensities within a patch.
- What is this going to be invariant to?

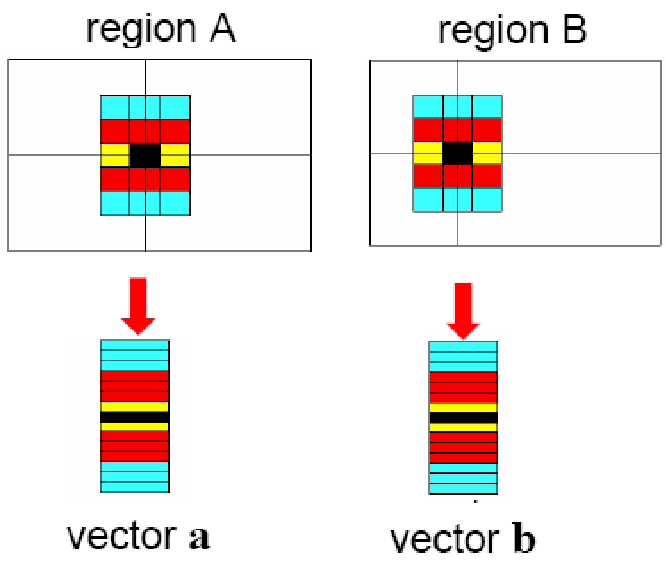
Write regions as vectors

$$\mathtt{A} \rightarrow \mathbf{a}, \;\; \mathtt{B} \rightarrow \mathbf{b}$$

$$\mathrm{NCC} = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$

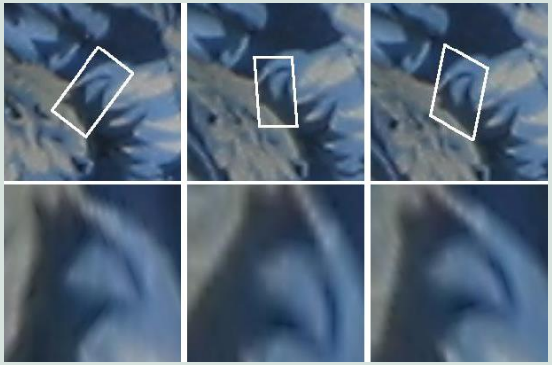
$$-1 \leq \mathrm{NCC} \leq 1$$



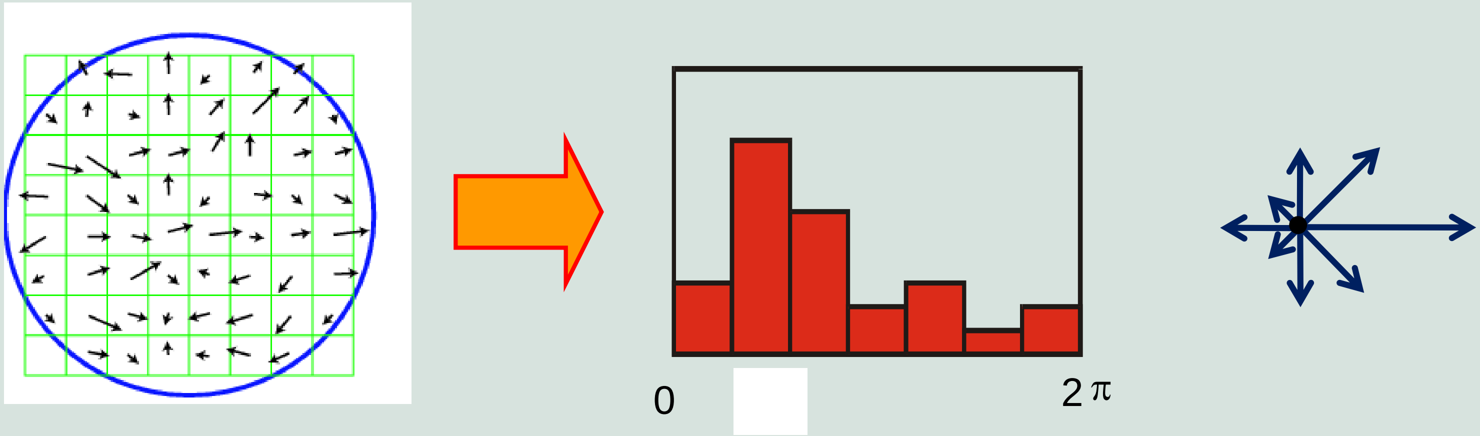


Feature Descriptors

- Disadvantage of patches as descriptors:
  - Small shifts can affect matching score a lot



- Solution: histograms



Detectors and Descriptors

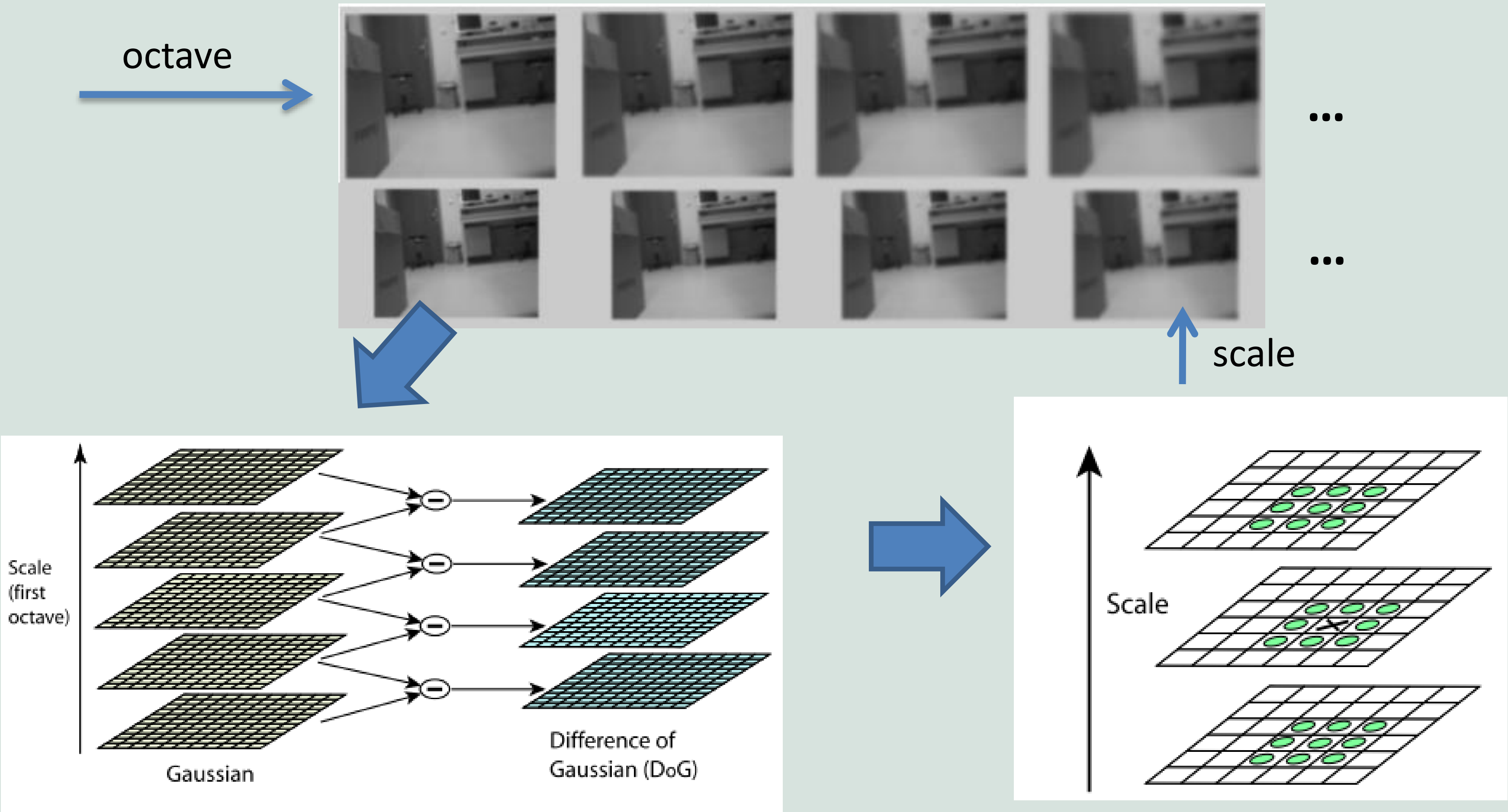
- Both detector and descriptor
  - SIFT (Scale Invariant Feature Transform)
    - Detector depends on LoG approximated by Difference of Gaussians (DoG)
    - Descriptor uses histograms of oriented gradients
  - SURF (Speeded-Up Robust Features)
    - Detector depends on determinant of Hessian (DoH)
    - Descriptor uses Haar-wavelets
  - ORB (Oriented FAST and Rotated BRIEF)
- Only descriptor
  - HoG (Histogram of Gradients)
    - depends on statistical information of gradients
  - LBP (Local Binary Patterns)
    - Depends on intensity variations in circular paths

SIFT ALGORITHM

- Extract features invariant to
  – Scale or orientation
  – Affine transformations and illumination (partially)
- Steps involve
  – Scale Space Extrema Detection
  – Keypoint Localization
  – Orientation Assignment
  – Keypoint Descriptor

Lowe (2004)

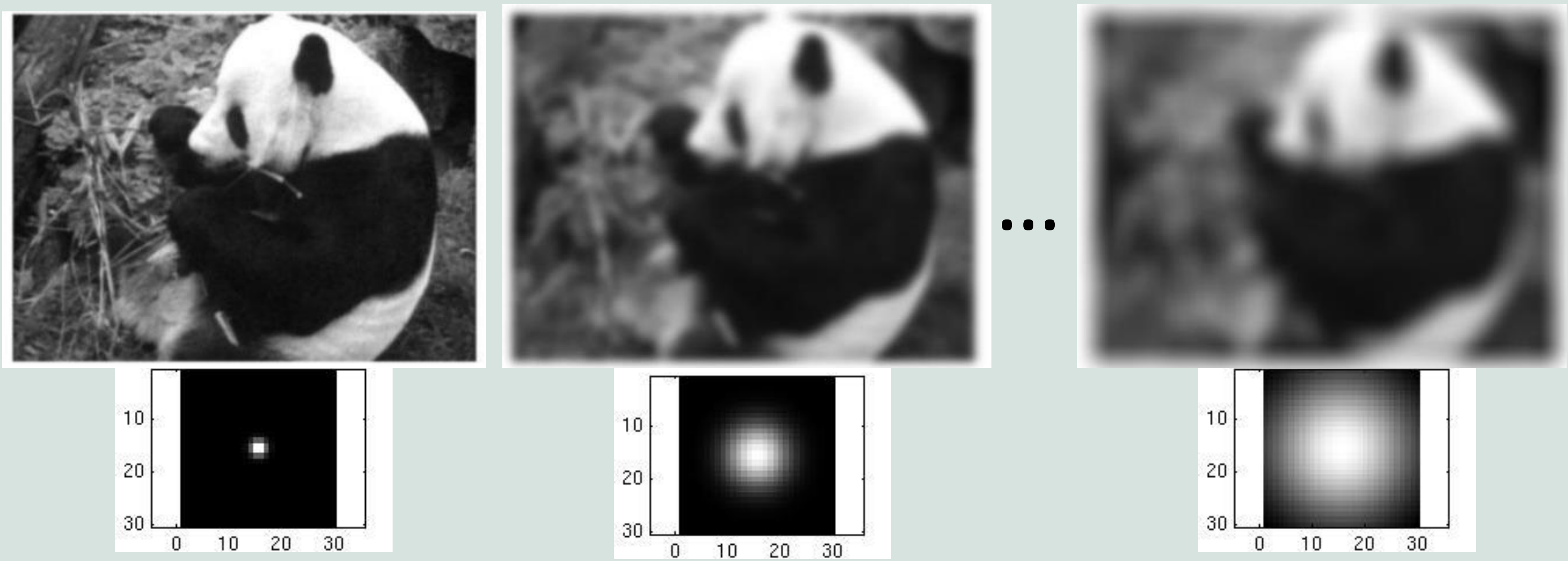
Scale Space Extrema Detection



- Starting from a σ_0 and using $\sigma_{i+1} = \sqrt{2}\sigma_i$, where i=0,1,2,3,…,7, compute $I^{\sigma_i} = I * G^{\sigma_i}$
- Halve or double the image size and repeat the above procedure.

Smoothing with a Gaussian

- Parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.



```
fig, ax = plt.subplots(nrows=1, ncols=4)
plt.gray()
for sig in range(1,11,3):
    out = snd.filters.gaussian_filter(im,sig)
    indexAxis = math.floor((sig-1)/3)+1
    plt.subplot(1,4,indexAxis), plt.imshow(out)
    plt.title((r'Gauss blur, $\sigma=$'+str(sig)))
plt.show()
```

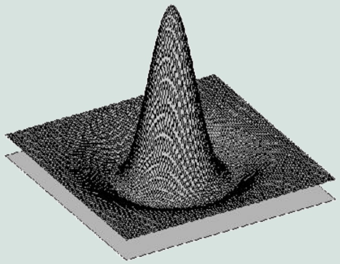
Effect of σ on Derivatives

- The apparent structures differ depending on Gaussian's scale parameter.
  - Larger values: larger scale edges detected
  - Smaller values: finer features detected

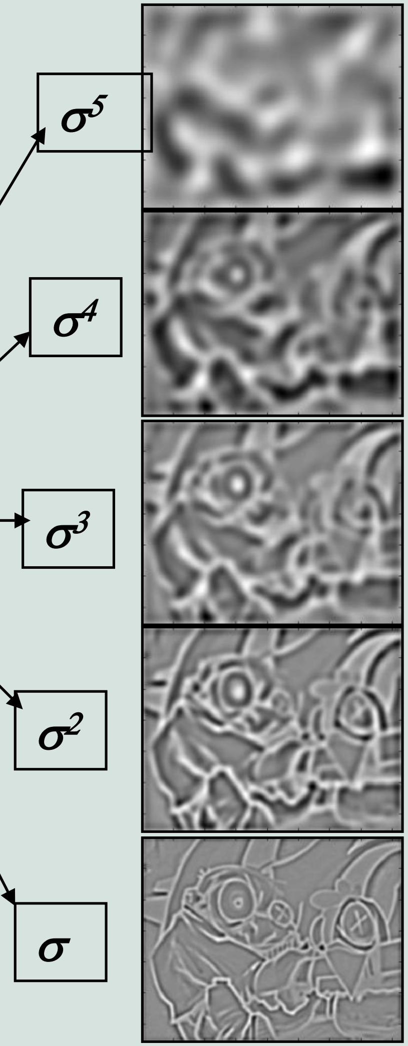
σ = 1 pixel

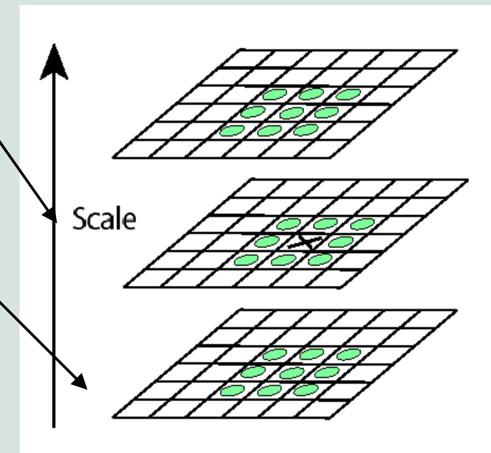
σ = 3 pixels

Laplacian-of-Gaussian (LoG)



$$L_{xx}(\sigma) + L_{yy}(\sigma)$$





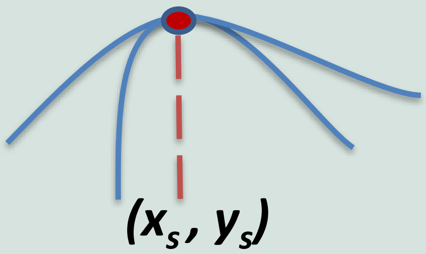
- Interest points are local maxima or local minima in scale spaces of either
  - Laplacian-of-Gaussian or
  - Difference-of-Gaussian

- 26 neighbourhood is checked to determine local maxima or minima

$\Rightarrow$ **List of *(x, y, σ)***

Keypoint Localization

- Pixels containing extreme values are approximate maxima and minima
- the interpolated (subpixel) locations of the maxima or minima are determined
- a 3D quadratic function is fit around each approximate extremum
- Extreme points can be found differentiating this function and equating it to 0.
  - Taylor series function where D and its derivatives are evaluated at (x, y, σ)
  - D(x)→Difference of Gaussian images
  - **x** = (x,y) coordinate of extremum
  - Maximum value of this quadratic function is obtained at subpixel location x_s, y_s



$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

Keypoint Localization

- A refinement procedure is followed to have more robust interest points
- Determinant of Hessian is used to eliminate edge responses and maintain corners.
  - L_{xx}, L_{xy}, *and* L_{yy} are second order derivative functions with respect to x or y
  - Refinement condition is checked using trace and determinant of Hessian

$$H(\sigma_i) = \begin{bmatrix} L_{xx}^{\sigma_i} & L_{xy}^{\sigma_i} \\ L_{yx}^{\sigma_i} & L_{yy}^{\sigma_i} \end{bmatrix} \qquad 0 \leq \frac{tr(H(\sigma_i))^2}{\det(H(\sigma_i))} \leq 12$$

Orientation Assignment

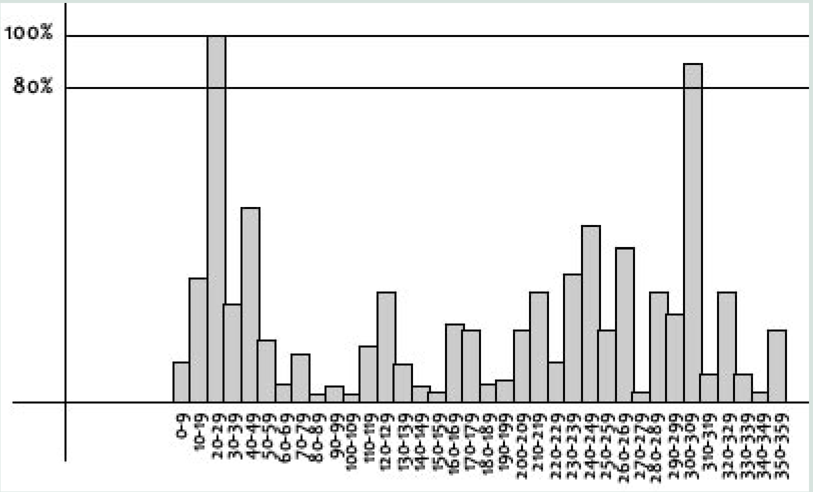
- In a local region around the keypoint, gradient magnitudes and directions are computed for each pixel
  - window size is typically 1.5 times the keypoint scale
- Histogram of oriented gradients is generated.
  - 36 bin histogram for better resolution
  - the most prominent one or more orientations in that region are assigned to the keypoint
  - the midpoint of the bin with greatest magnitude is selected

Magnitude:

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

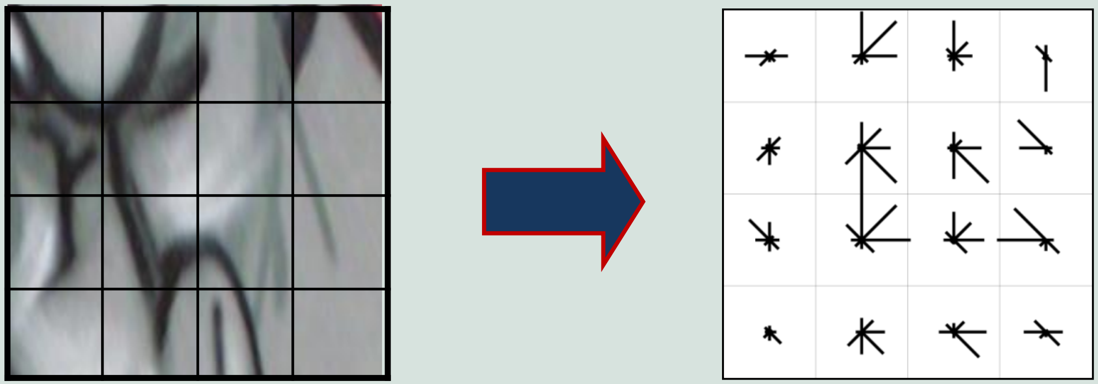
Orientation:

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



Keypoint Descriptor

- A way to distinguish each keypoint from others uniquely
- Again in a window resized according to the keypoint scale, compute the descriptor
  - the image gradient magnitudes and orientations are sampled around the keypoint location
  - the neighbours are Gaussian weighted and window is divided into 4x4 cells
  - Histogram of gradients are computed for each cell using 8 bins and normalized
  - the descriptor coordinates and the gradient orientations are rotated relative to the keypoint orientation



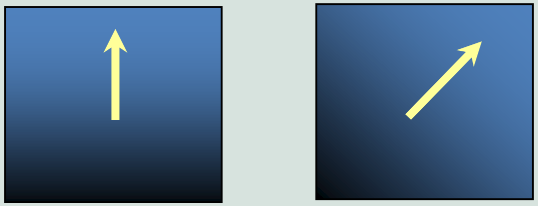
In SIFT:
The window is split into 16 (4×4) cells

Keypoint descriptor length:
16 cells x 8 bins = 128

Rotation Invariant Descriptors

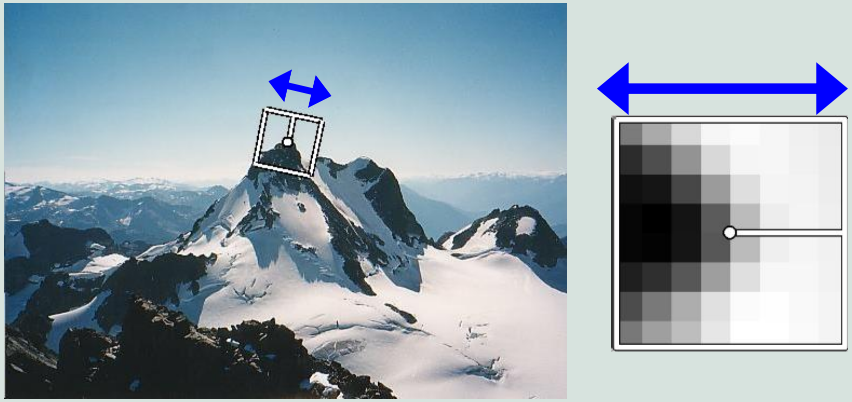
- Find local orientation

  Dominant direction of gradient for the image patch



- Rotate patch according to this angle

  This puts the patches into a canonical orientation.



Scale Invariant Feature Transform (SIFT)

- Describes image features that are
  - invariant to
    - scale
    - rotation
  - partially invariant to
    - change in illumination
    - 3D camera viewpoint
    - occlusion, clutter
- Fast and efficient—can run close to real time
- Lots of code available
- Provides a basis for object and scene recognition.

Object Recognition with SIFT

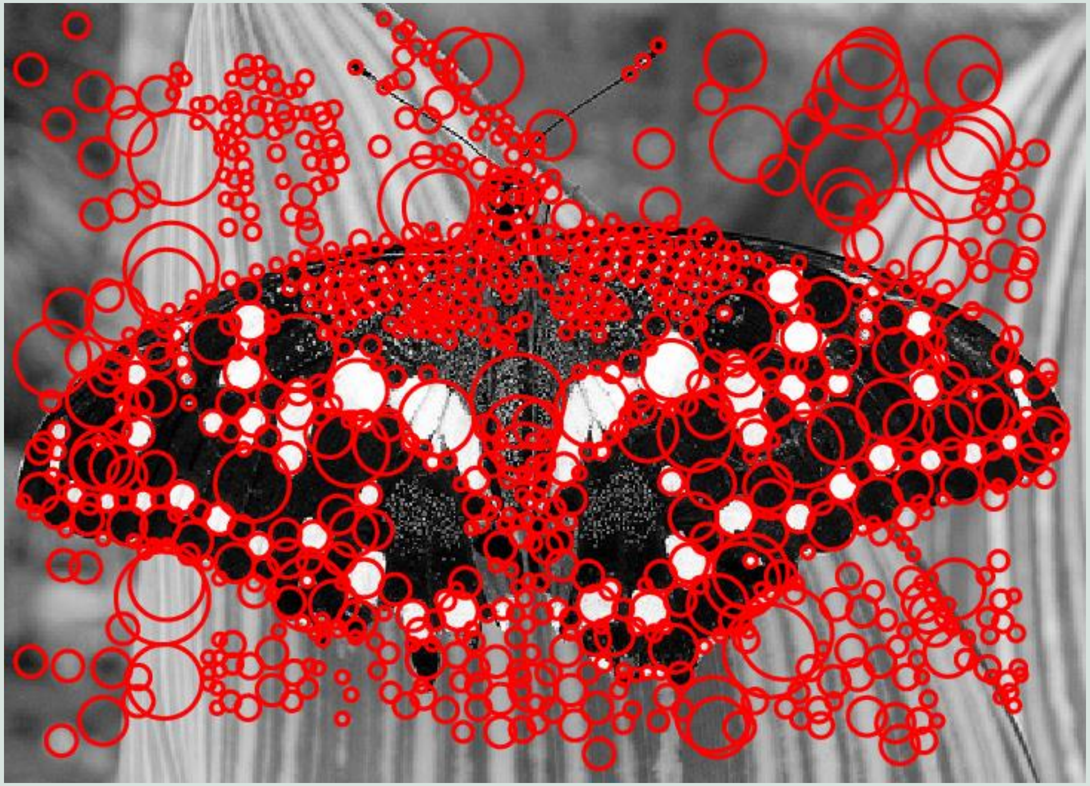
A parallelogram -> each recognized object's boundaries
Smaller squares -> the keypoints that were used for recognition.

Scale-Space Blob Detector: Example

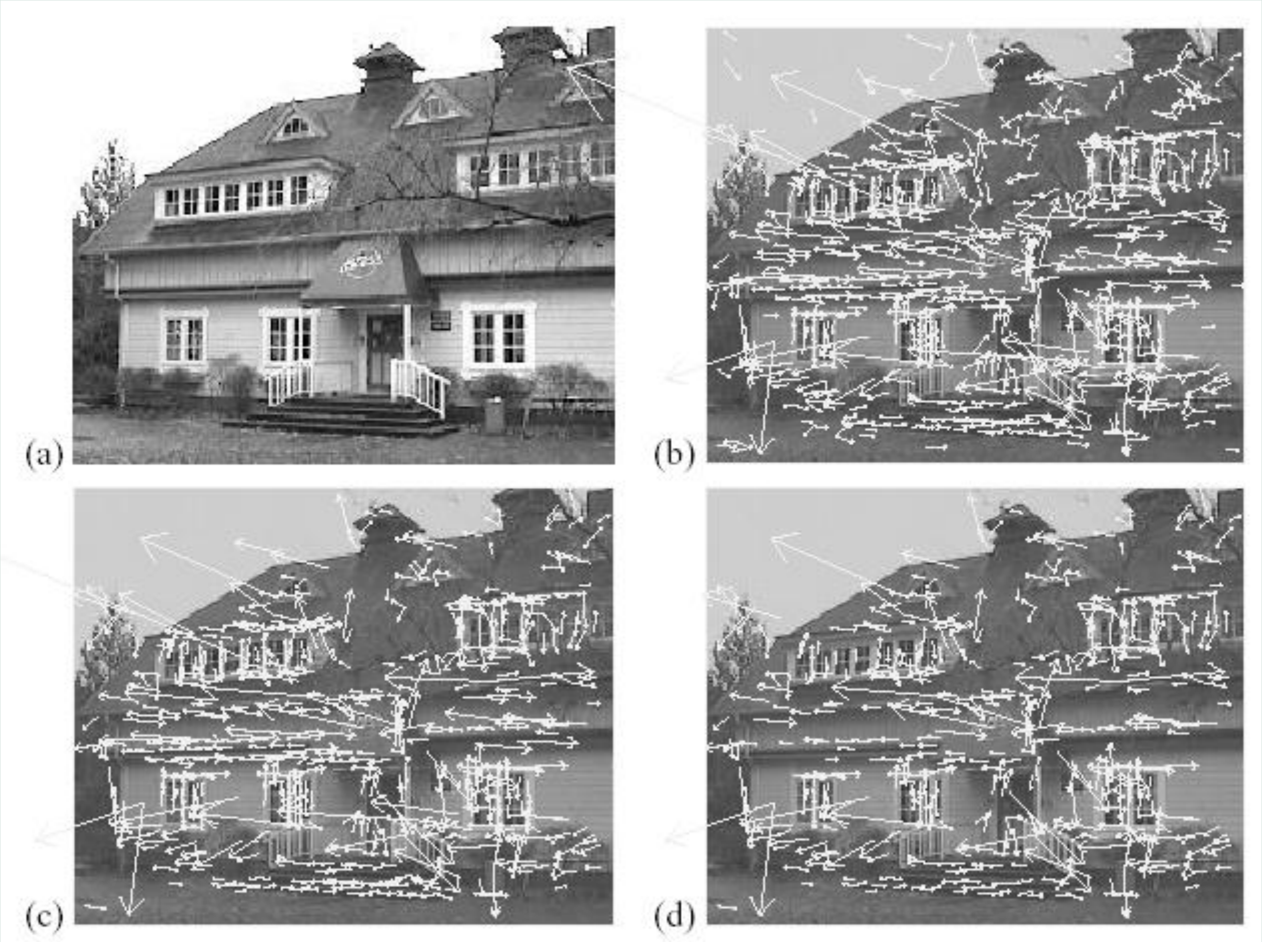
Scale-Space Blob Detector: Example

sigma = 9.5859

Scale-Space Blob Detector: Example



Key Point Detection: Example



(a) 233x189 image

(b) 832 DOG extrema

(c) 729 left after peak value threshold

(d) 536 left after testing ratio of principle curvatures (removing edge responses)

Working with SIFT Descriptors

- One image yields:
  - n 128-dimensional descriptors: each one is a histogram of the gradient orientations within a patch
    - [n x 128 matrix]
  - n scale parameters specifying the size of each patch
    - [n x 1 vector]
  - n orientation parameters specifying the angle of the patch
    - [n x 1 vector]
  - n 2d points giving positions of the patches
    - [n x 2 matrix]

- Remember each patch is related to a keypoint.

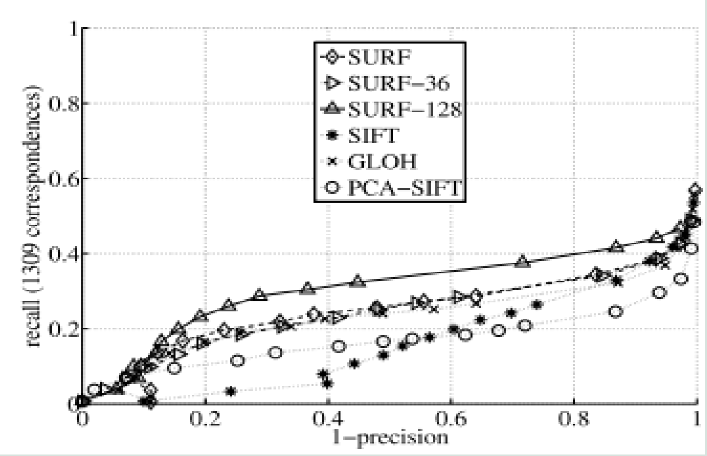
Speeded-up Robust Features (SURF)

- a scale and rotation invariant interest point detector and descriptor
- Method:
  - Detector was based on the Hessian matrix but relied on the integral images to reduce the computation time
    - Hessian matrix→ a matrix of second order derivatives
  - The descriptor described the distribution of Haar-wavelet responses within the interest point neighborhood
  - Generally only 64 dimensions were used reducing the time for feature computation and modeling.

Bay et al. (2006)

Speeded-up Robust Features (SURF)

- Results:
  - sometimes more than 10% improvement in recall for the same level of precision
  - For the same number of points, computations of the detector and the descriptor
    - 391 ms for SURF-128
    - 1036 ms for SIFT
- approximated or outperformed previously proposed approaches respect to
  - repeatability
  - distinctiveness and robustness
  - computation and comparison time

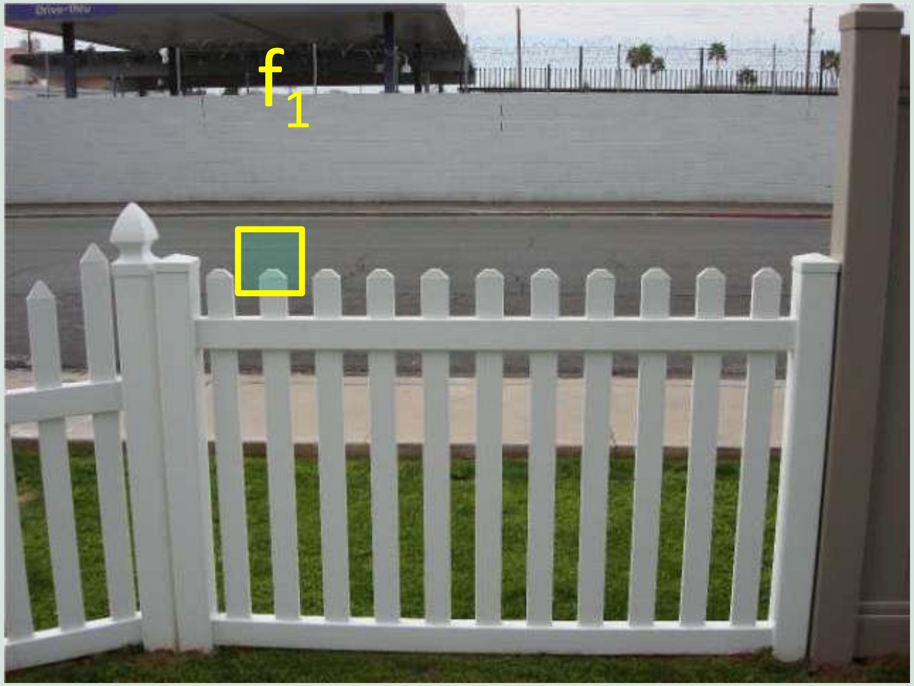


Feature Matching

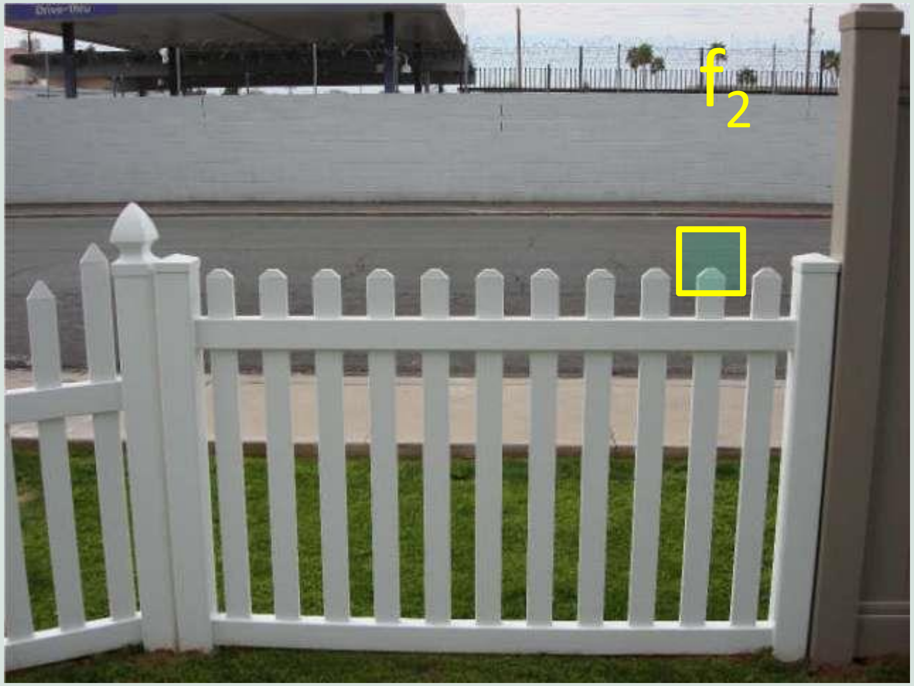
- Given a feature in I_1, how to find the best match in I_2?
  1. Define distance function that compares two descriptors
  2. Test all the features in I_2, find the one with min distance

Feature Distance

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



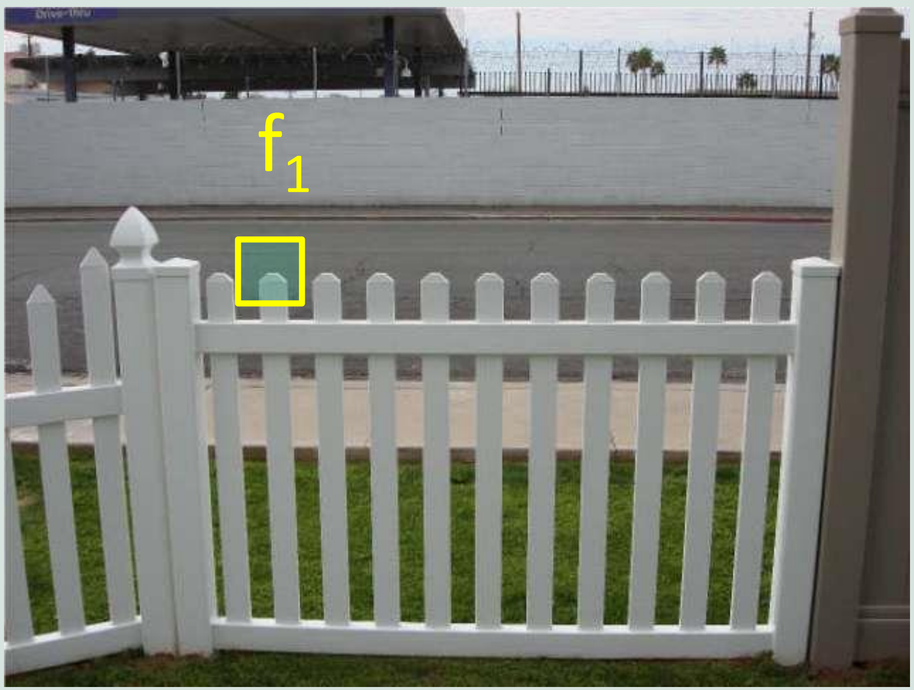
I_1



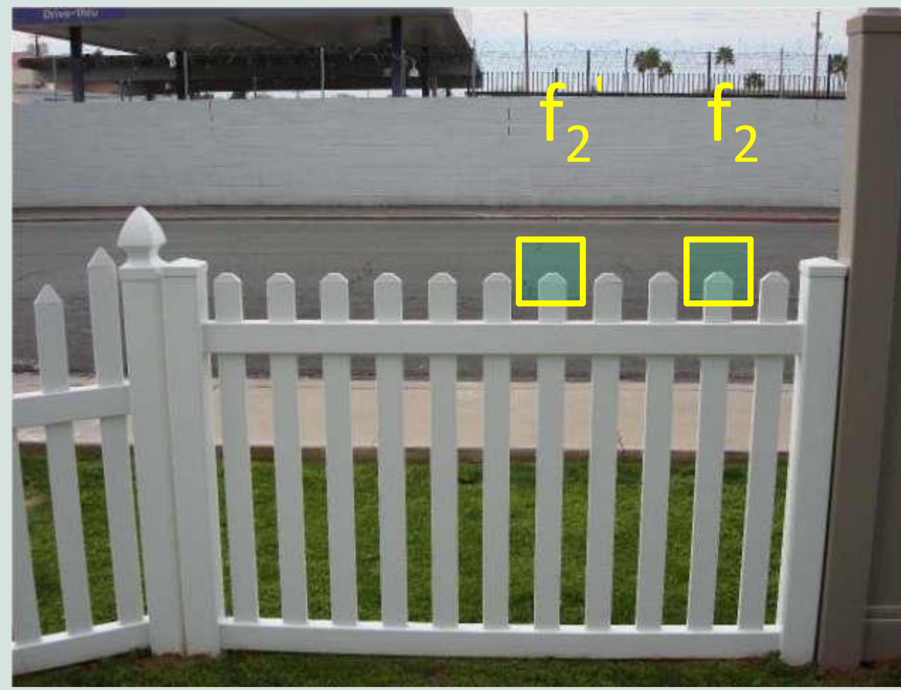
I_2

Feature Distance

- How to define the difference between two features f_1, f_2?
  - 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 2^{nd} best SSD match to f_1 in I_2
    - gives large values for ambiguous matches



I_1



I_2

Matching Keypoints

SIFT

SURF

Feature Detector and Descriptors Summary

- Stable (repeatable) feature points can be detected regardless of image changes
  - Scale: search for correct scale as *maximum* of appropriate function
  - Affine: approximate regions with *ellipses* (this operation is affine invariant)
- Invariant and distinctive descriptors can be computed
  - Using invariant *moments*
  - *Normalizing* with respect to scale and affine transformation

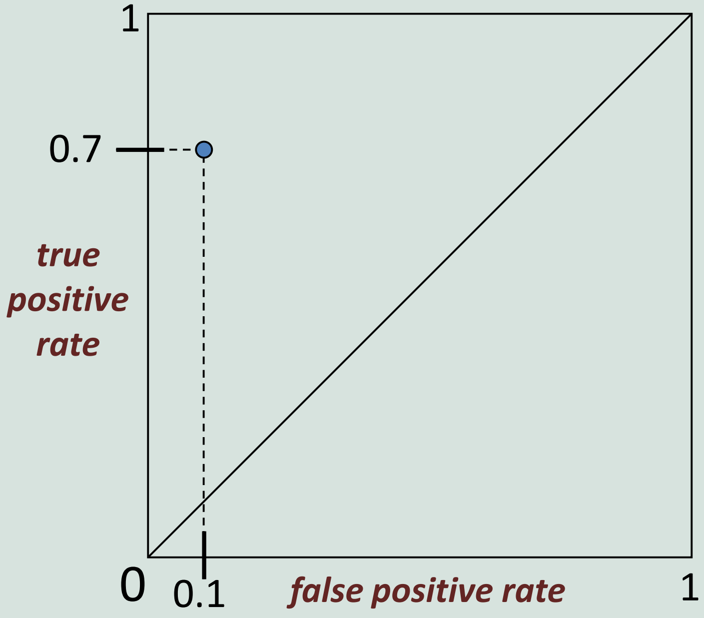
Evaluating the Matching Results

- 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 Matching Results

- How can we measure the performance of a feature matcher?

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



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

True positive rate (recall) = $\frac{TP}{TP+FN}$

False positive rate = $\frac{FP}{FP+TN}$

Precision = $\frac{TP}{TP+FP}$

F1-score = $\frac{2*Precision*Recall}{Precision+Recall} = \frac{2TP}{2TP+FP+FN}$

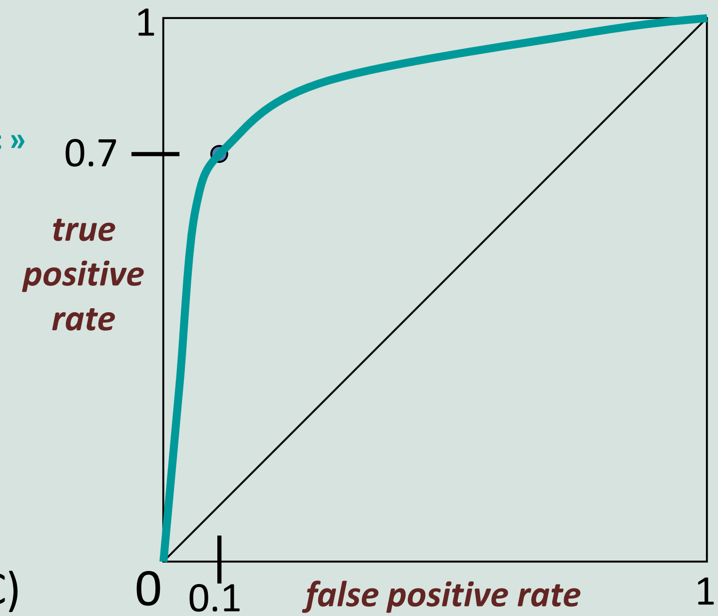
Evaluating the Matching Results

- How can we measure the performance of a feature matcher?

ROC curve

« Receiver Operator Characteristic »

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



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

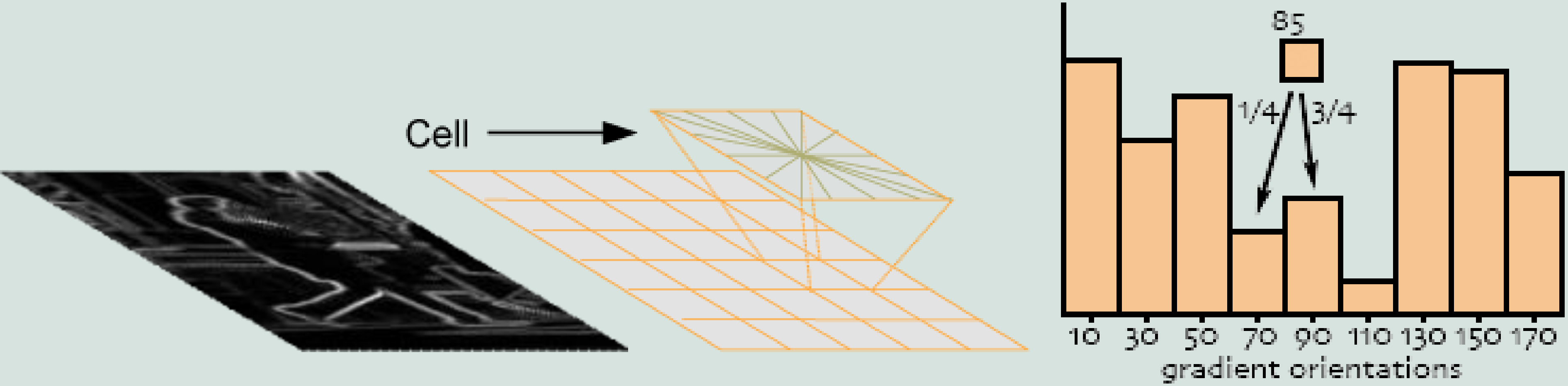
- ROC Curves
  - Generated by counting # correct matches / # incorrect matches, for different thresholds
  - Want to maximize area under the curve (AUC)
  - Useful for comparing different feature matching methods

Histogram of Oriented Gradients (HOG)

- Compress image to 64x128 pixels
- Convolution with [-1 0 1] [-1;0; 1] filters
- Compute gradient magnitude + direction
- For each pixel
  - take the color channel with greatest magnitude as final gradient

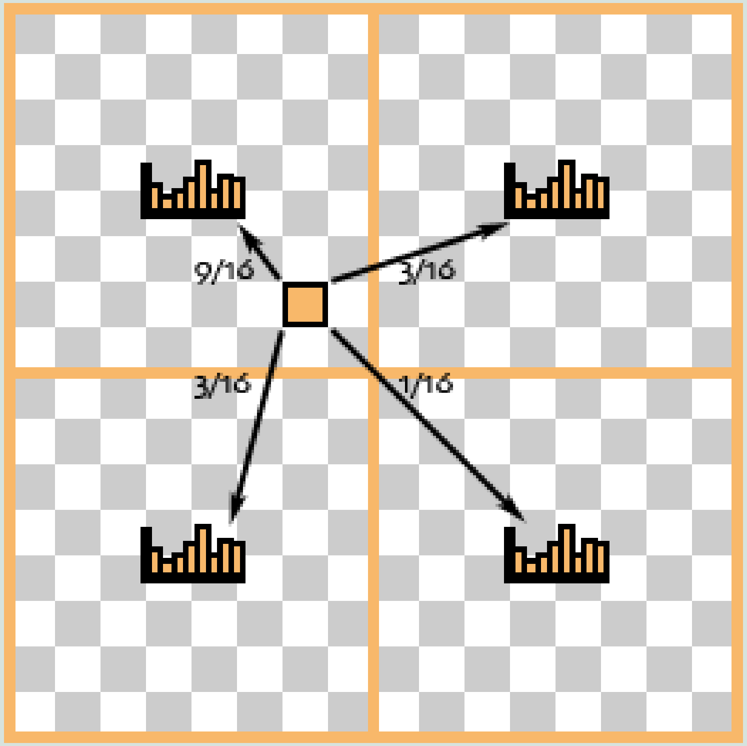
HoG: Cell Histograms

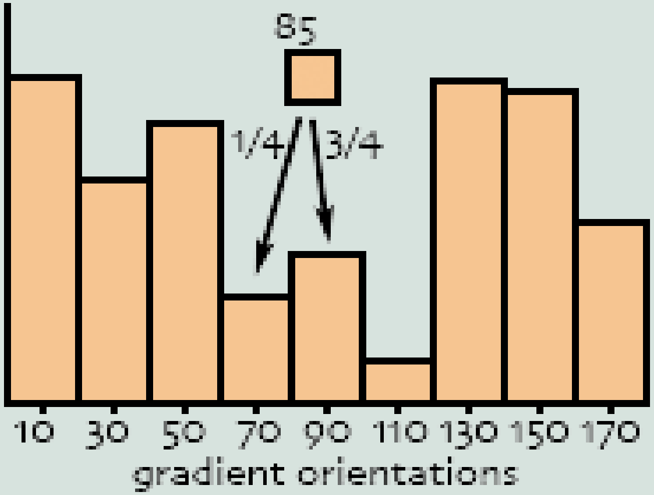
- Divide the image to cells, each cell 8x8 pixels
- Snap each pixel's direction to one of 18 gradient orientations
- Build histogram per-cell using magnitudes



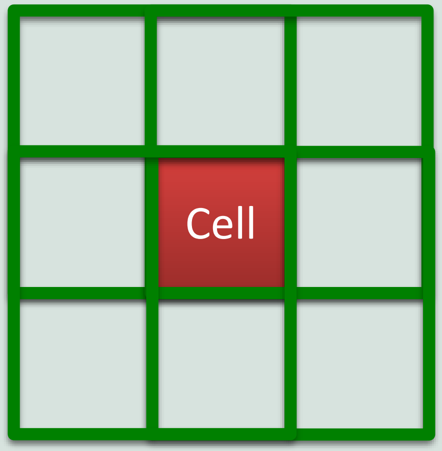
HoG: Histogram Interpolation

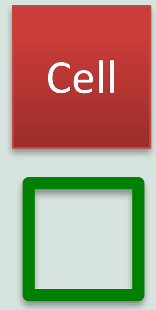
- Interpolated trilinearly:
  - Bilinearly into spatial cells
  - Linearly into orientation bins





HoG: Normalization





Current cell : 1x18 histogram

Block: 2x2 cell
overlapping with current cell

1. **contrast sensitive features:** 18 orientation -> 18 dim
2. **contrast insensitive features:** 9 orientation -> 9 dim

   Normalize 4 times by its neighbor blocks, and average them

3. texture features: sum of the magnitude over all orientation and normalize 4 times by its neighbor blocks, not average -> 4 dim

In total each cell : 18+9+4 dimensions of feature

Final Descriptor

- Concatenation the normalized histogram

Visualization:

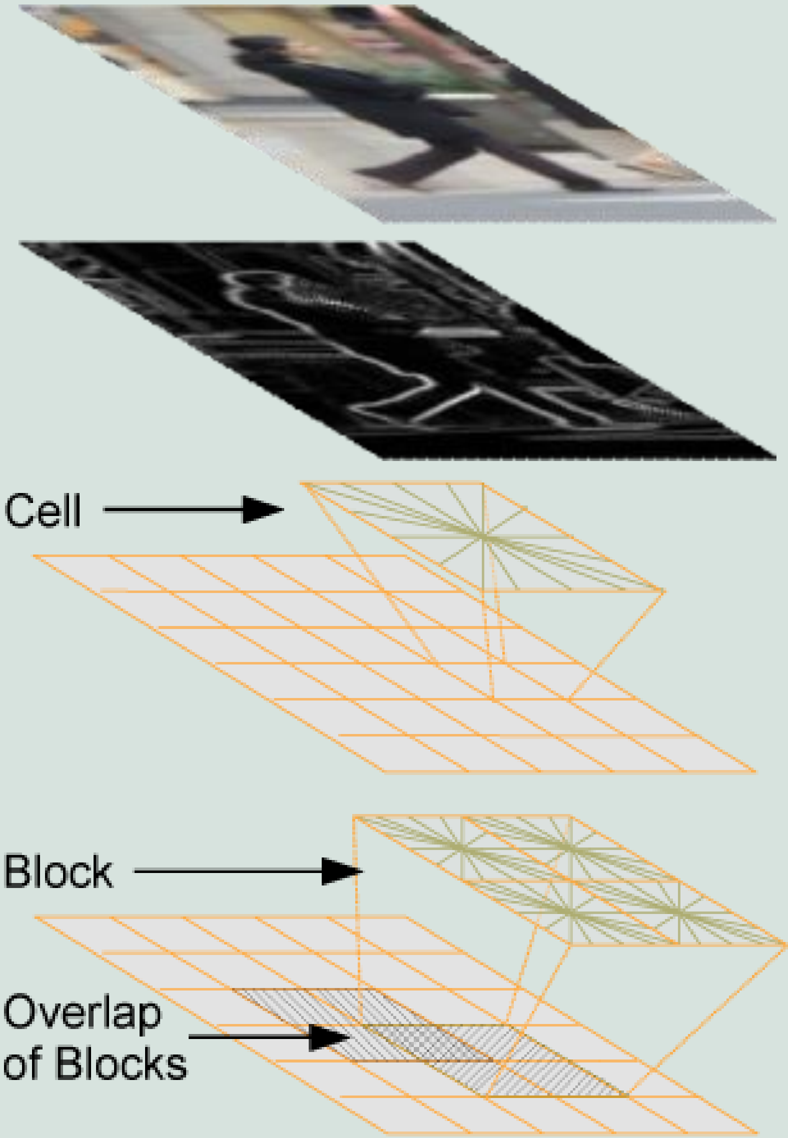
HOG Descriptor:

1. **Compute gradients** on an image region of 64x128 pixels

2. **Compute histograms** on «cells» of typically 8x8 pixels (i.e. 8x16 cells)

3. **Normalize histograms** within overlapping blocks of cells

4. **Concatenate histograms**



It is a typical procedure of feature extraction !

Local Binary Patterns

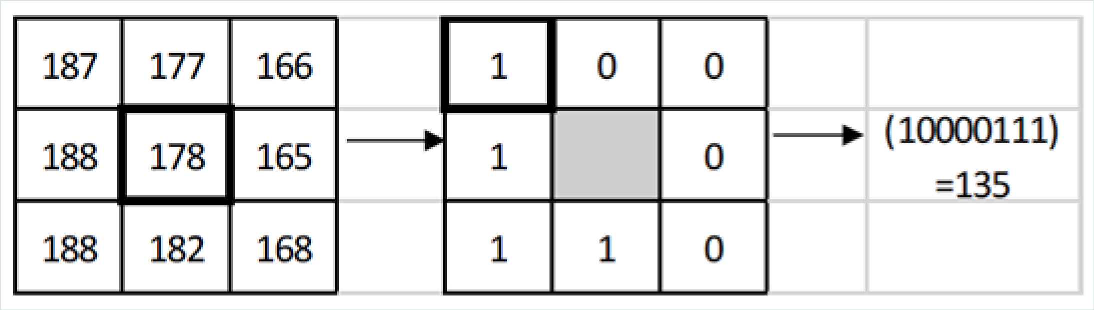
- A non-parametric feature extraction method (Ojala v.d., 1994, 1996)
- Analyzes texture pattern and gives orthogonal measures of local contrast
- LBP is
  – Highly tolerant to different illumination conditions
  – Sensitive to small changes in gray-level image
  – Low cost to compute

LBP Algorithm

- Input image

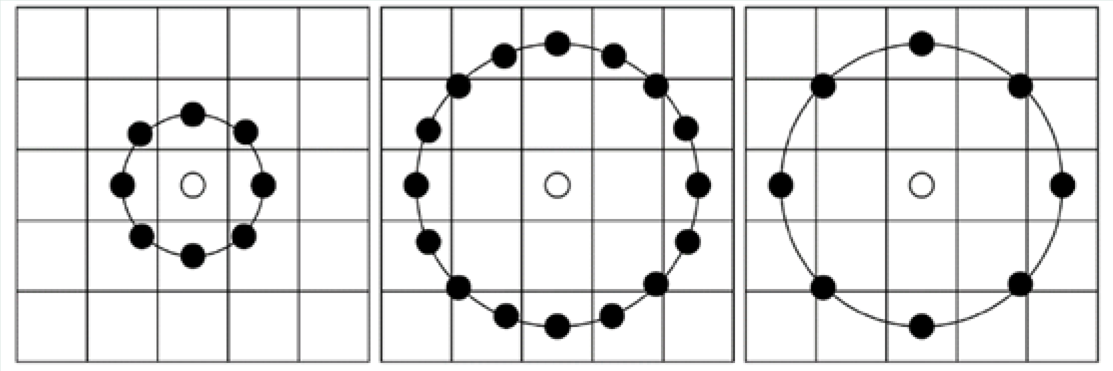
1. Divide image into cells (e.g. every cell has 16×16 pixels)
2. Every pixel in a cell is compared to its neighbors
   - Neighborhood can be in (8, R) or (16, R) structure where R is distance to neighbors and 8 or 16 is the number of neighbours.
   - Follow pixels along a circular path in clockwise or counter clockwise direction
   - If central pixel is greater than or equal to neighbor , label as 0 otherwise label as 1
   - As a result for 8 neighborhood an 8-digit binary number is obtained
   - This binary number is generally converted to a decimal number
3. A frequency histogram of the decimal numbers is computed in each cell
   - A 256–length feature histogram is obtained
4. 256–length feature histogram is converted to 59-length uniform pattern
5. Normalized histograms of each cells is combined to have a feature vector for the whole image

LBP Neighborhood



$$LBP_{P,R}(x_c) = \sum_{p=0}^{P-1} u(x_p - x_c)2^P, \quad u(y) = \begin{cases} 0, & \text{if } y < 0 \\ 1, & \text{if } y \geq 0 \end{cases}$$

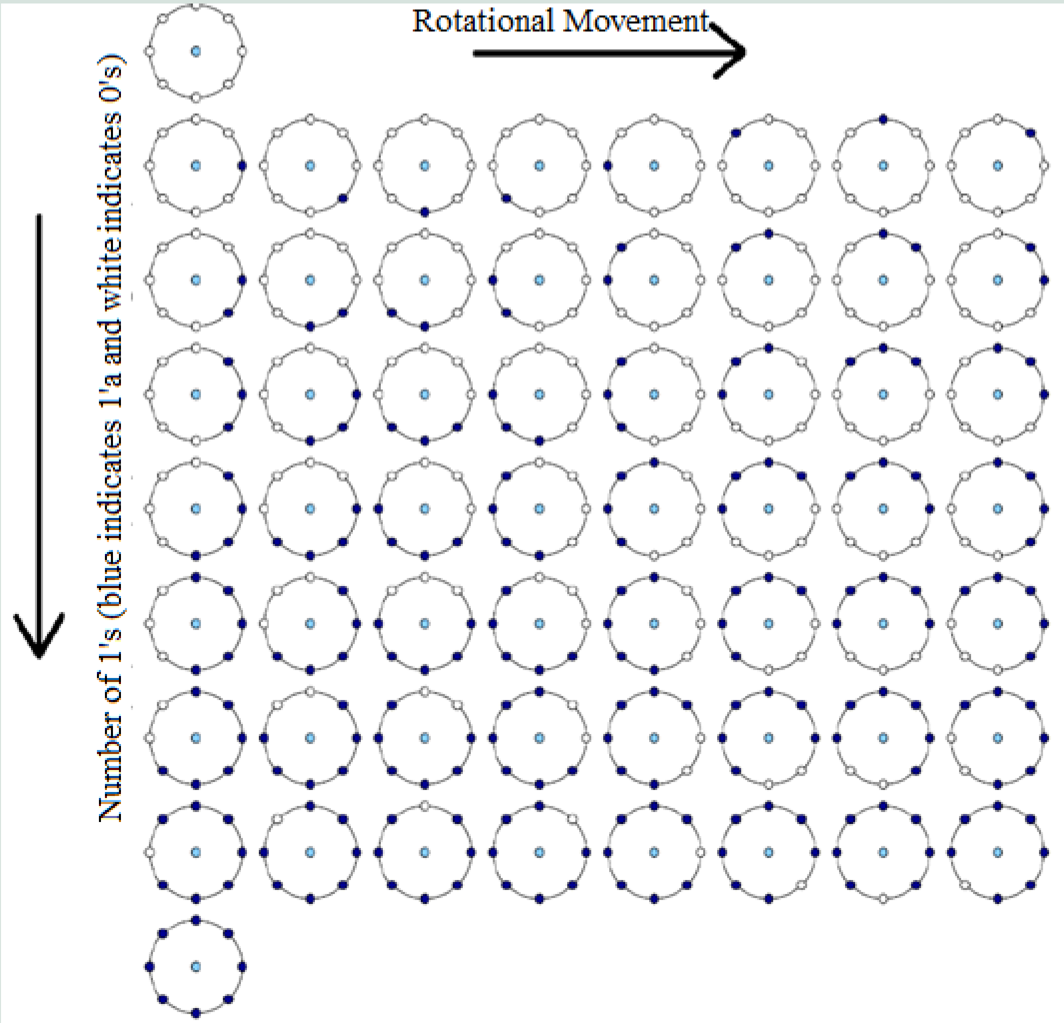
- Circular (8, 1), (16, 2) and (8, 2) neighborhoods



LBP Patterns

- For a P-bit binary number there are 2^P different LBP codes
  - 8 bit number → 2^8=256 different patterns
  - LBP patterns can be
    - uniform
    - non-uniform
- Uniform LBP patterns
  - An extension of LBP
  - more robust to noise and computationally less complex
  - yield better detection or recognition results
    - also related to rotational invariance issue
  - If number of passes from «0» to «1» less than or equal to two
    - 0000000 (0 pass) →uniform
    - 00011100 (2 passes) →uniform
    - 11101111 (2 passes)) →uniform
    - 00110100 (4 passes) →nonuniform
    - 01010100 (6 passes)) →nonuniform

58 Different Uniform Patterns for (8, R) Neighborhood

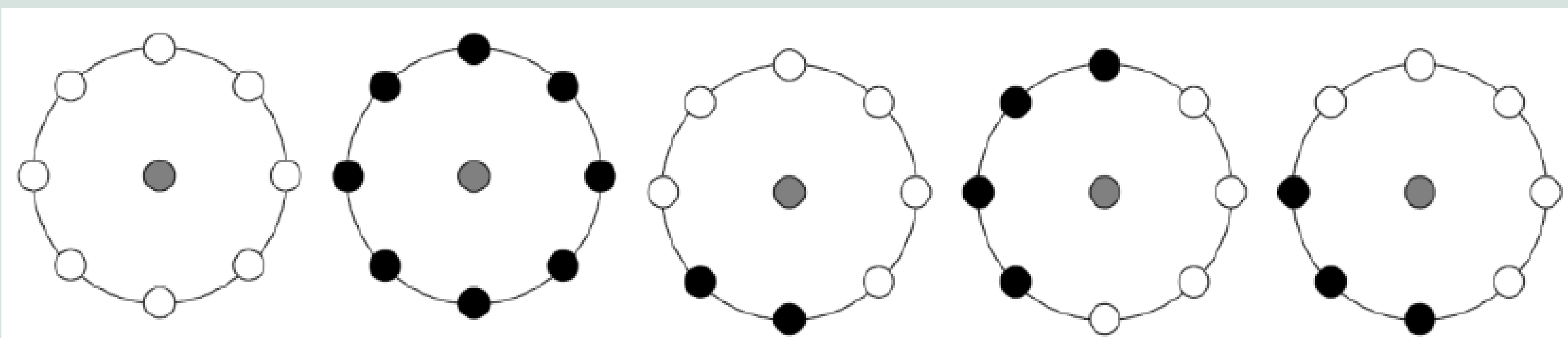


- Every uniform pattern has a different label
- Every nonuniform pattern assigned to a single label
- For 8-bit pattern
  - 59 labels

(Pietikäinen et al., 2011)

Some Structures Detected by LBP

- Black→bit 1
- White→bit 0
- Gray→central pixel



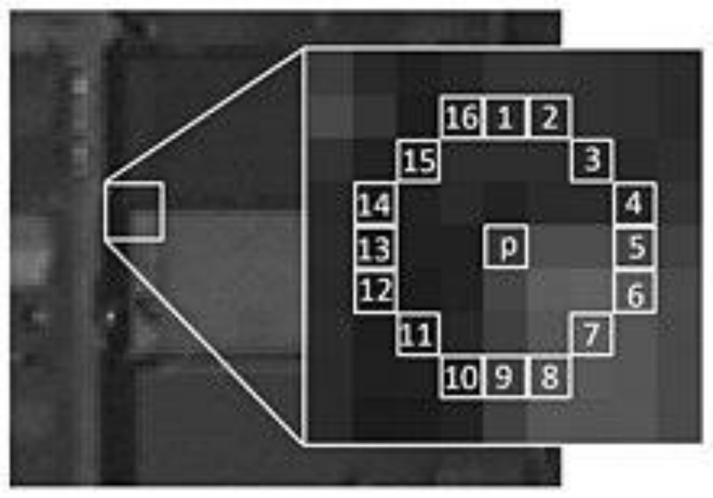
Spot Flat Line end Edge Corner

Oriented FAST and Rotated BRIEF (ORB)

- Combines FAST feature detectors and BRIEF descriptors, (Ethan Rublee, 2011)
- Enables fast detections and descriptions of features
  - Useful for real-time applications such as robotics vision systems or smartphone applications

Features from Accelerated Segment Test (FAST)

- Uses pixel neighborhood to compute key points in an image
- Algorithm
  - An interesting point candidate pixel (i,j) is selected with an intensity I(i,j)
    - In a circle of 16 pixels, given a threshold t,
    - estimate n adjoining points which are brighter or darker than pixel (i,j) by the threshold t.
    - n is chosen as 12
  - In a high-speed test only four pixels at 1, 9, 5, 13 in the figure are checked. Intensity values of at least three of these pixels determine whether (i,j) is a corner.



FAST Features with Orientation

- FAST does not produce rotation information
- ORB uses FAST features with orientation information
- Using a circular radius of 9 pixels a vector between the computed intensity centroid and center of the corner is used to describe orientation
- In circular region the moments are computed

$$m_{p,q} = \sum_{x,y} x^p y^q I(x, y)$$

- Using computed moments intensity centroid is determined

$$C = \left(\frac{m_{1,0}}{m_{0,0}}, \frac{m_{0,1}}{m_{0,0}}\right)$$

- Patch center O and intensity centroid C are joined to form the orientation vector $\overrightarrow{OC}$.
- The orientation of the patch then simply is:

$$\theta = \mathrm{atan2}(m_{0,1}, m_{1,0})$$

Binary Robust Independent Elementary Features (BRIEF)

- A descriptor proposed by Michael Calonder et al. (2010)
- Unlike relatively high dimensional features of
  - SIFT: 128 x 4 byte
  - SURF: 64 x 4 byte
  - BRIEF consumes less memory
    - 128, 256 or 512 bits
    - 16, 32, 64 bytes
- Similar to LBP, BRIEF computes intensity differences in a small patch of image and represents them as a binary string

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