MAEG 5720:
Computer Vision in Practice

Lecture 5:

Shape Descriptor

Dr. Terry Chang

2021-2022

Semester 1

香港中文大學
The Chinese University of Hong Kong

Department of Mechanical and Automation Engineering
機械與自動化工程學系

Last Lecture

- Edge Detector
  - Gradient Detector
    - Roberts
    - Prewitt
    - Sobel
  - Laplacian of Gaussian (Marr-Hildreth)
  - Gradient of Gaussian (Canny)
- Interest Point Detection
- Scale-invariant Region Selection

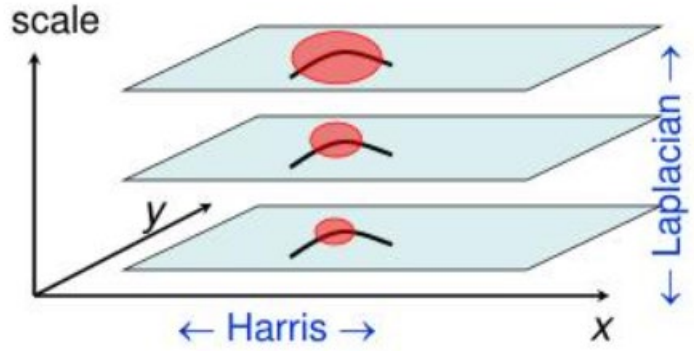
Recall: Harris Corner Detector

- Harris :
  - Detect ***Corners*** using second moment matrix
  - Invariant to ***rotation***, ***partial luminance*** change
  - ***Not*** Invariant to ***scale*** change

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

$$R = \det(M) - k\big(trace(M)\big)^2$$

- Harris-Laplace :
  - Detect ***Corners*** using Harris
  - Detect ***Maximum responds*** using LOG
  - Invariant to ***rotation*** and ***scale change***



Today's Agenda

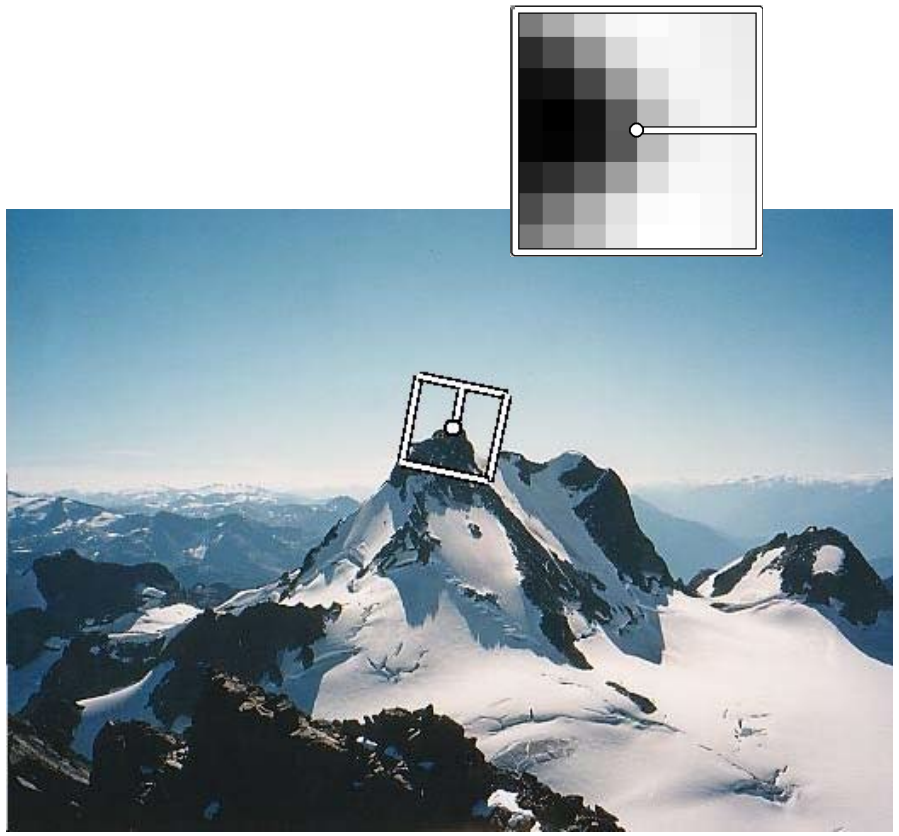
- Shape Descriptor
  - Scale Invariant Feature Transform (SIFT)
  - Speed Up Robust Feature (SURF)
  - Oriented FAST and Rotated BRIEF (ORB)
  - Histogram of Oriented Graph (HOG)

- Visual Bags of Words.

A Window Feature Descriptor

- How can we represent the ***local area*** around each point for ***matching purpose***?

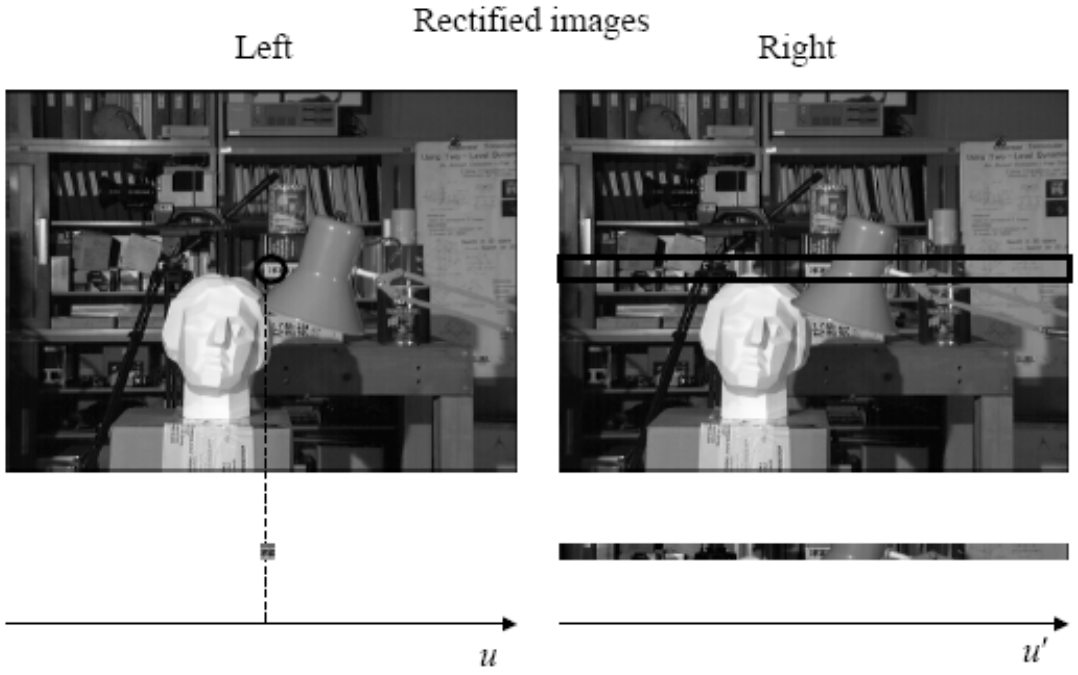
- The simplest way is to consider a list of luminance of a $k \times k$ window around the ***feature point*** to form a *feature vector*



150
76
161
33
49
87
240
233
152
88
61
214
88
127
231
130
161
128
70
204
164
199
98
162
…

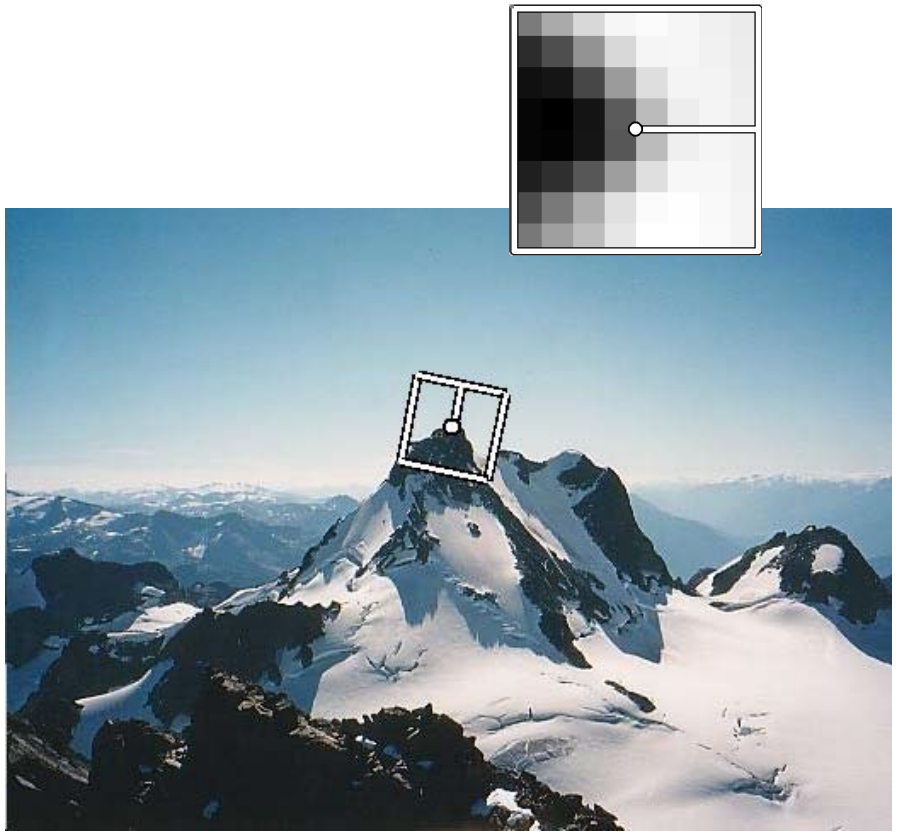
Simple approach: correlation

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



Problem

- What if the scale changes?
- What if the orientation changes?
- What if the luminance of all pixel changes?
- Can we do better?



Need a robust feature descriptor

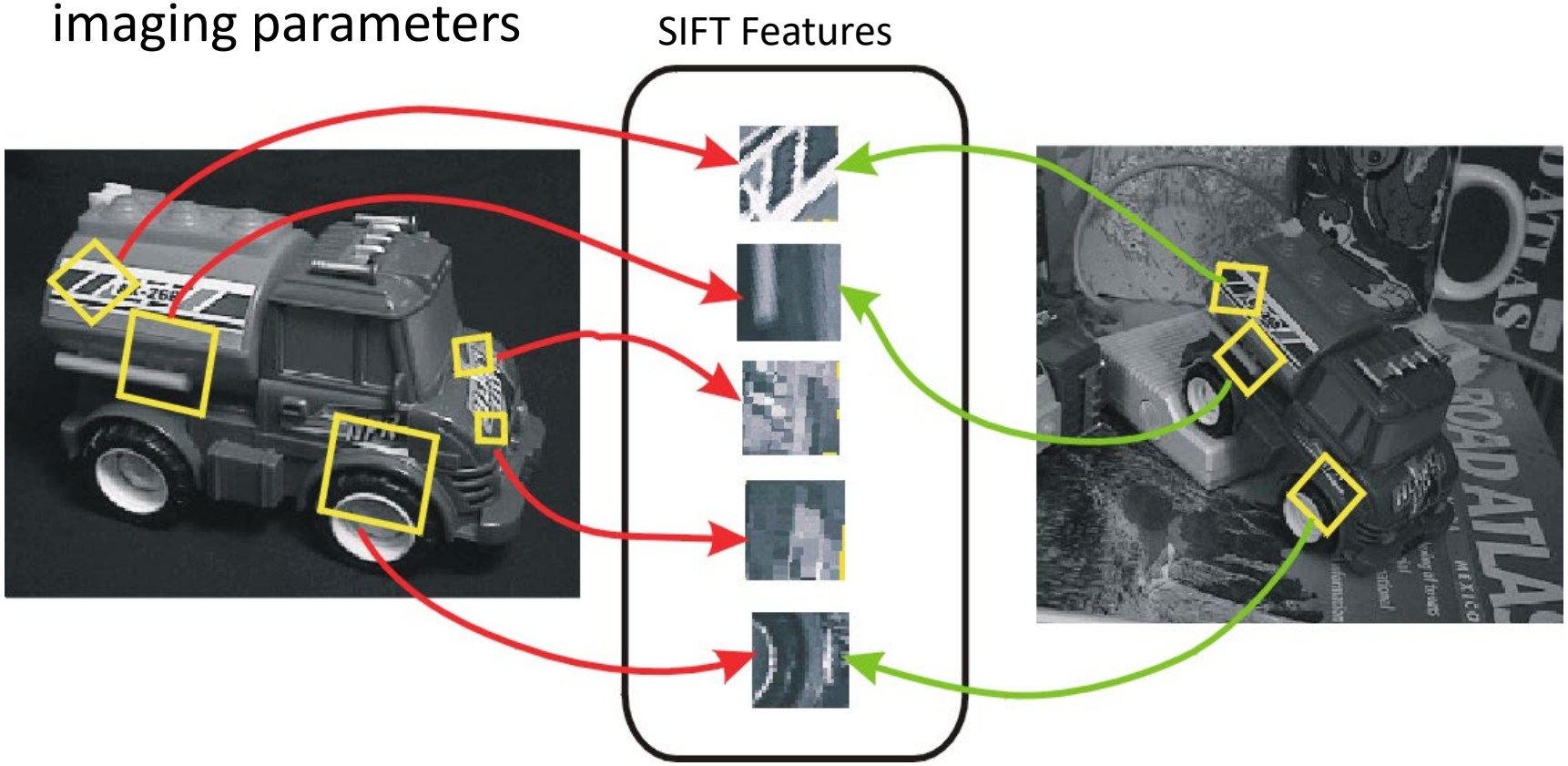
- SIFT: Scale Invariant Feature Transform (***SIFT***) Lowe 2004

  Transform Image data into scale-invariant coordinates relative to local features.

  - How to describe those point for matching?
    - Invariant to
      - Scale
      - Rotation
      - Illumination
      - Deformation

Invariant Local Features

- Image content is transformed into ***local feature coordinates*** that are invariant to ***translation***, ***rotation***, ***scale***, and other imaging parameters



Slides adopted from David Lowe **CVPR 2003 Tutorial Recognition and Matching Based on Local Invariant Features**

Advantages of invariant local features

- **Locality**: features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness**: individual features can be matched to a large database of objects
- **Quantity**: many features can be generated for even small objects
- **Efficiency**: close to real-time performance
- **Extensibility**: can easily be extended to wide range of differing feature types, with each adding robustness

Scale invariance

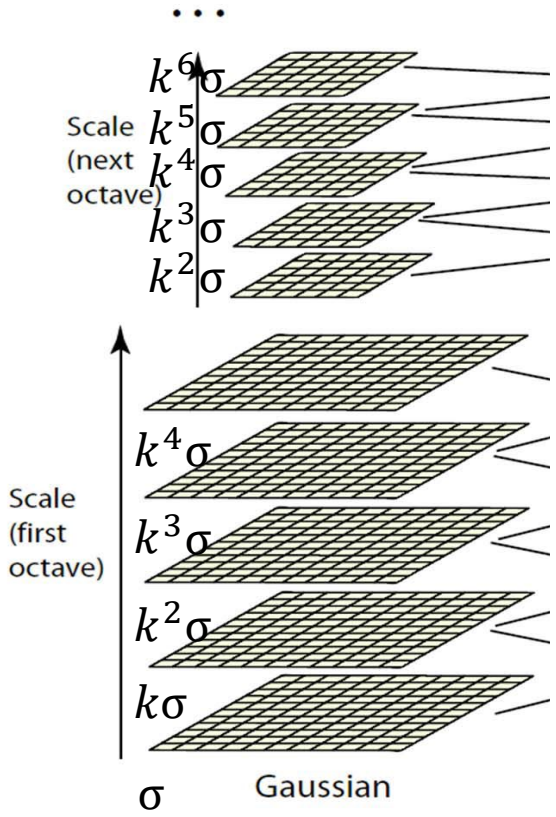
Requires a method to repeatably select points in location and scale:

- The only reasonable scale-space kernel is a Gaussian (Koenderink, 1984; Lindeberg, 1994)
- An efficient choice is to detect peaks in the difference of Gaussian pyramid (Burt & Adelson, 1983; Crowley & Parker, 1984 – but examining more scales)
- Difference-of-Gaussian with constant ratio of scales is a close approximation to Lindeberg's scale-normalized Laplacian (can be shown from the heat diffusion equation)

Summarizes the steps for building SIFT Descriptor

1. *Constructing a scale space* - Building the smoothing versions of the image with different σ value and image sizes (octaves).
2. *LoG Approximation by DoG* – Approximate LoG by DoG which is less computational expensive.
3. *Finding key points* - Search for maxima/minima in DoG images across scales
4. *No maxima suppression* – Eliminate the low contrast key points and edges
5. *Assigning an orientation* – The orientation of each pixel is calculated, and canonical orientation is defined at the peak. All pixels' orientation are defined based on the canonical orientation. The orientation is quantiles into 8 orientations(bins) and histogram is constructed.
6. *Generate SIFT feature description* – generate the 8-bin histogram of 4x4 regions making ***128 dimensions descriptor***.

1. Building a Scale Space

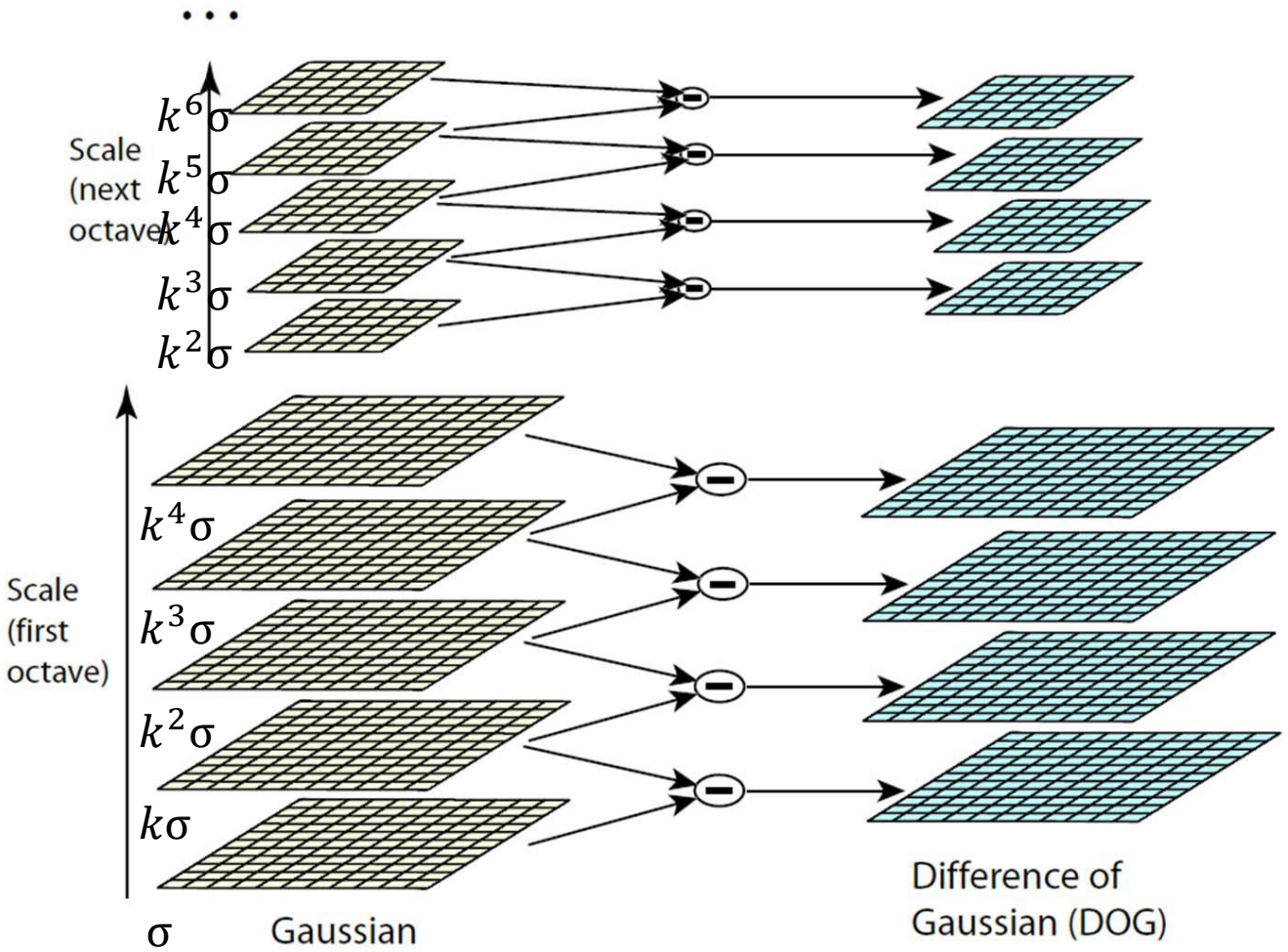


- All scale must be examined to identify scale-invariant features
- An efficient function is to compute the ***Laplacian Pyramid*** (by Difference of Gaussian) (Burt & Adelson, 1983)
- An octave corresponds to doubling the value of σ
- Fixed number of scales per octave

2. LoG approximation by DoG

$$G(x, y, k\sigma) = \frac{1}{2\pi(k\sigma)^2} e^{-\frac{x^2+y^2}{2k^2\sigma^2}}$$

Down-sample



$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

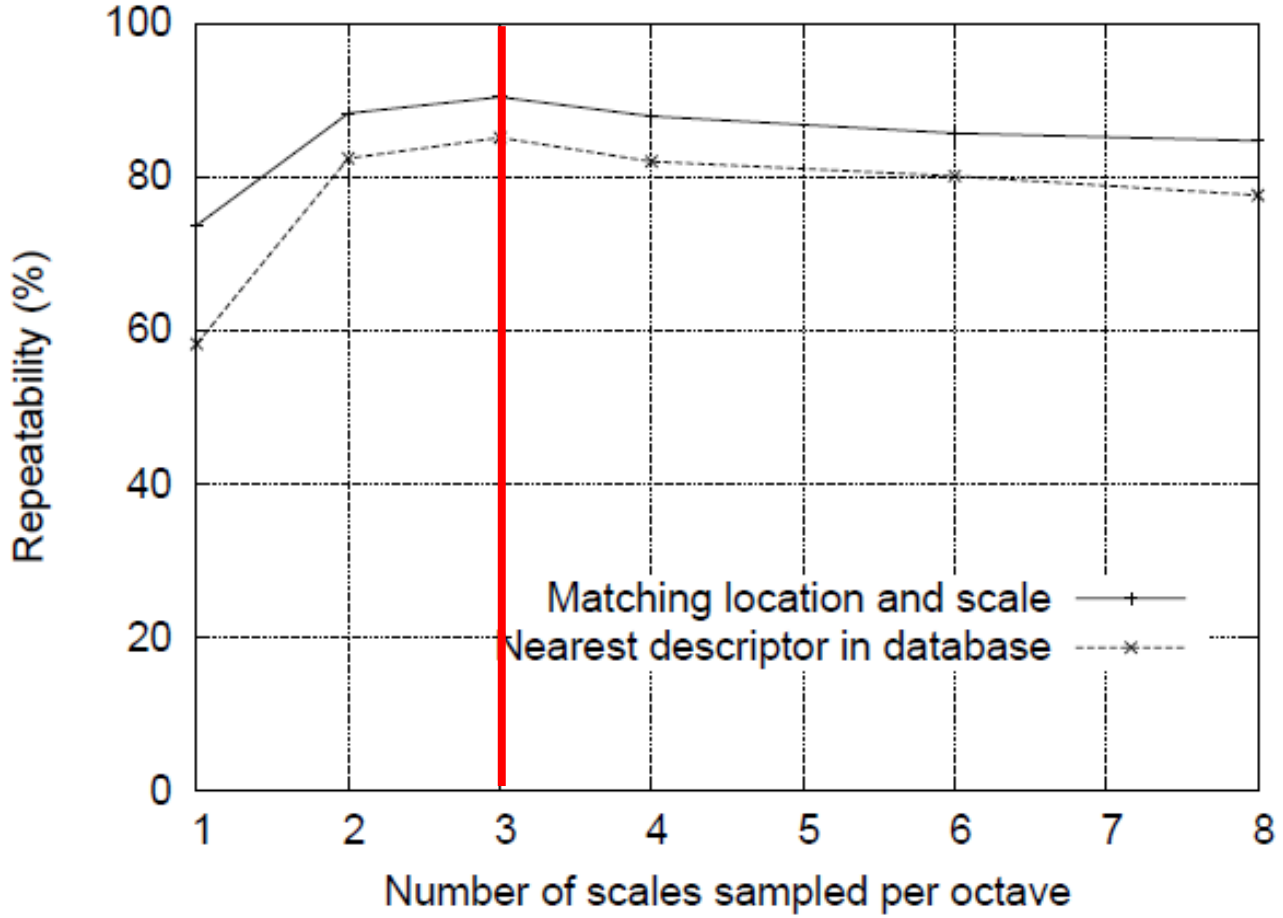
$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

$$k = \sqrt{2}$$

Choosing SIFT parameters

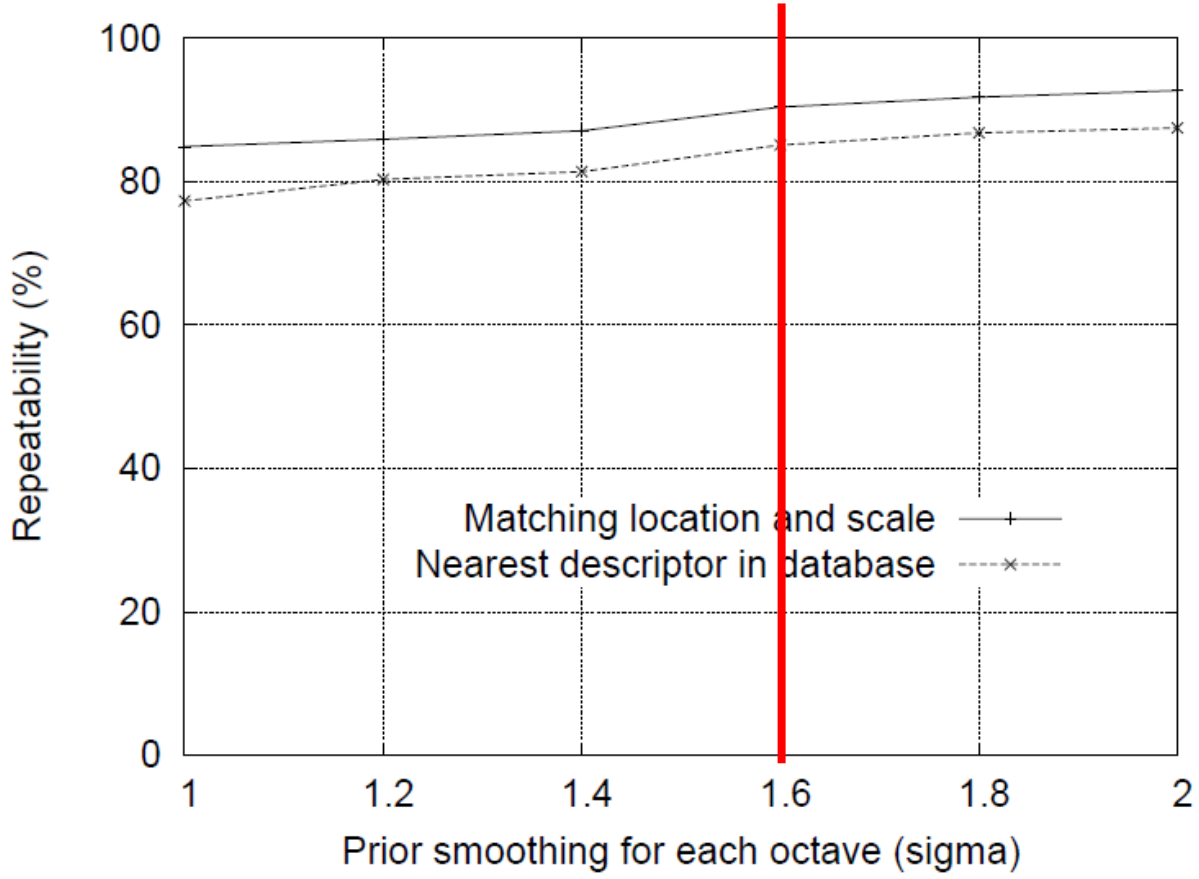
- Experimentally using a matching task:
  - 32 real images (outdoor, faces, aerial etc.)
  - Images subjected to a wide range of transformations (i.e., rotation, scaling, shear, change in brightness, noise).
  - Keypoints are detected in each image.
  - Parameters are chosen based on keypoints repeatability, localization, and matching accuracy.

How many scales per octave?



Source: D. Lowe 04

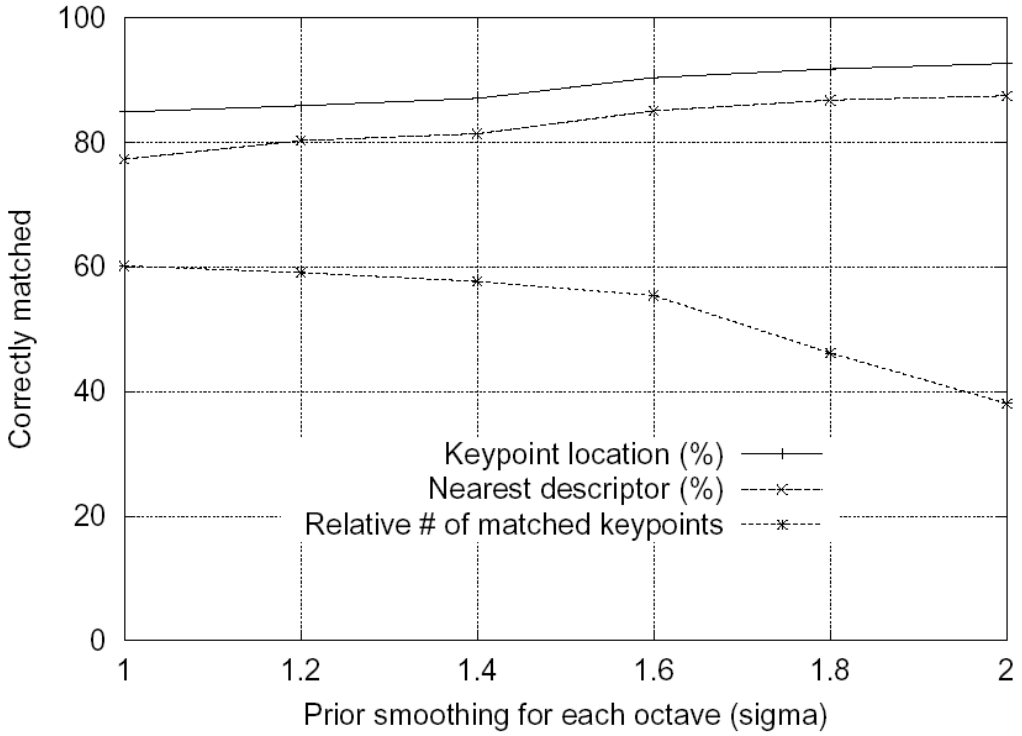
Initial Sigma Value



Sampling frequency for spatial domain

Need to determine number of pixels (samples) relative to smoothing

- Image size is doubled to keep highest spatial frequencies

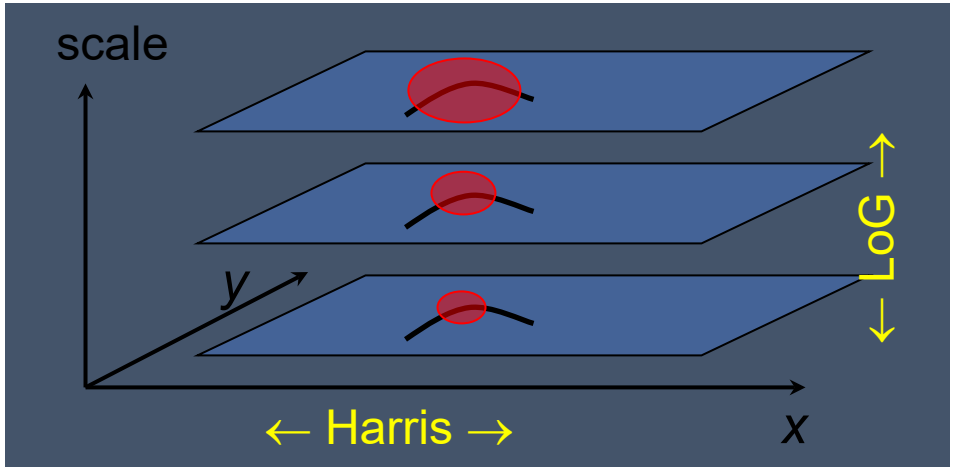


3. Keypoint detection and localization

Harris-Laplacian

Find local maxima of:

- Harris detector in space
- LoG in scale



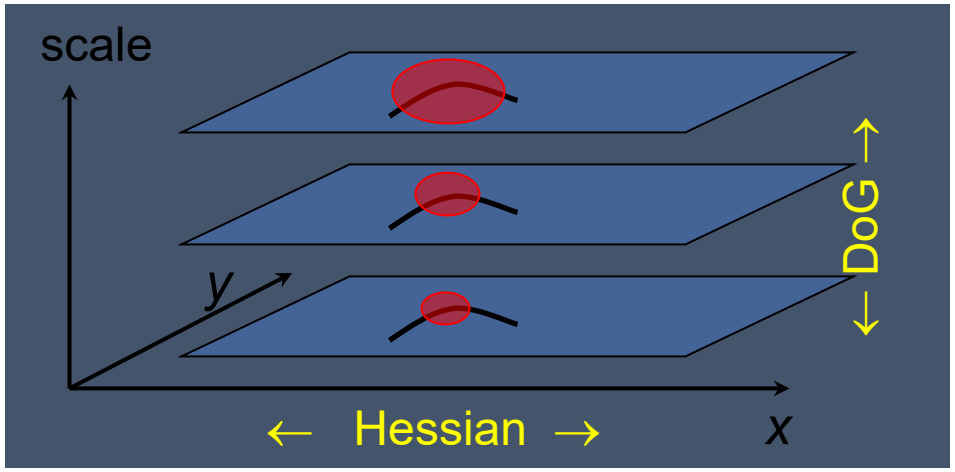
- SIFT

Find local maxima of:

- Hessian in space
- DoG in scale

$\sigma_n = k^n \sigma_0$

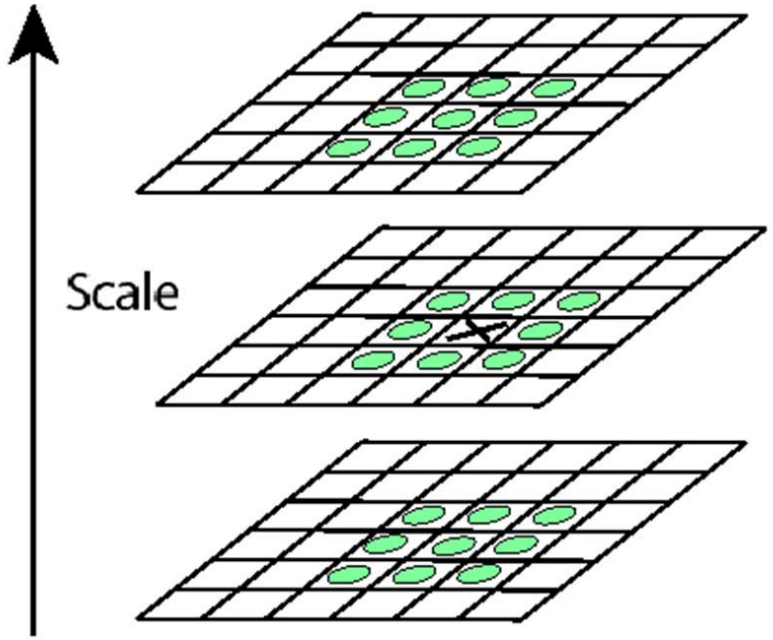
$(k=2)$



Slide credit: George Bebis

3. Key point detection and localization in scale space

- Detect Maxima of difference-of-Gaussian (DoG) in scale space
- Compare a pixel (X) with 26 pixel in current and adjacent scales
- Select pixel (X) if it is Maxima

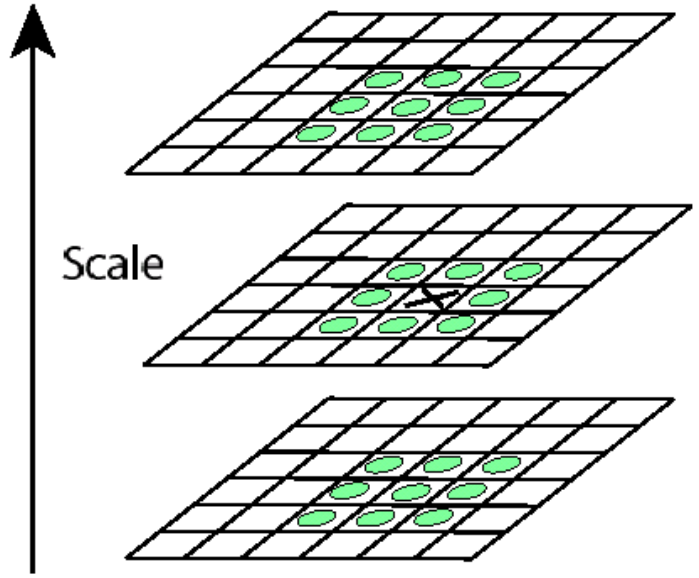


Candidate key points list of
(x,y,σ)

Slide credit: David Lowe

Keypoint localization - Sub-pixel accuracy

- Determine the location and scale of keypoints to **sub-pixel** and **sub-scale** accuracy by fitting a quadratic least square fit at each keypoint.
  - $D(x, y, \sigma) \rightarrow D(x', y', \sigma')$



- The 2nd Taylor expansion of D at (x, y, σ):

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

Take the derivatives with respect to $\overrightarrow{\Delta x}$

$$\left(\frac{\partial D}{\partial \Delta x}\right) = \left(\frac{\partial D}{\partial x}\right) + \left(\frac{\partial^2 D}{\partial x^2}\right)(\Delta x)$$

Keypoint localization - Sub-pixel accuracy

Solve for $\left(\frac{\partial D}{\partial \Delta x}\right) = 0$ to find the extremum

$$(\Delta x) = -\left(\frac{\partial^2 D}{\partial x^2}\right)^{-1} \cdot \left(\frac{\partial D}{\partial x}\right)$$

Where $(\Delta x) = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \sigma \end{bmatrix}$, $\frac{\partial D}{\partial x} = \begin{bmatrix} \frac{\partial D}{\partial x} \\ \frac{\partial D}{\partial y} \\ \frac{\partial D}{\partial \sigma} \end{bmatrix}$, $\frac{\partial^2 D}{\partial x^2} = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial x \partial \sigma} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} & \frac{\partial^2 D}{\partial y \partial \sigma} \\ \frac{\partial^2 D}{\partial x \partial \sigma} & \frac{\partial^2 D}{\partial y \partial \sigma} & \frac{\partial^2 D}{\partial \sigma^2} \end{bmatrix}$

Extermum (x, y, σ) moved to $(x + \Delta x, y + \Delta y, \sigma + \Delta \sigma)$

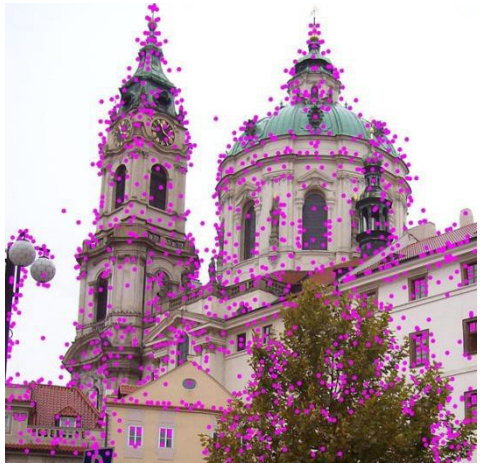
Repeat if Δx, Δy, or $\Delta \sigma > 0.5$

4. Non maximum suppression

- Threshold
  - Low contrast candidates
  - Apply Threshold to maxima

$D(x', y', \sigma') > threshold$ (0.03-Lowe)

After scale space extremas are detected



SIFT algorithm discards low contrast keypoints

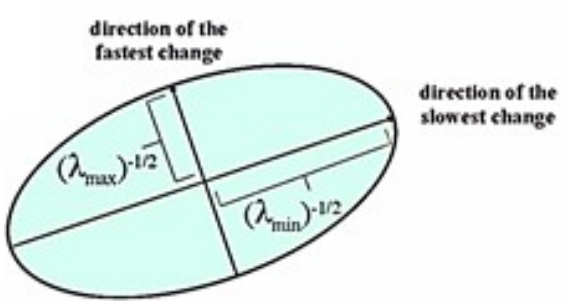
http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

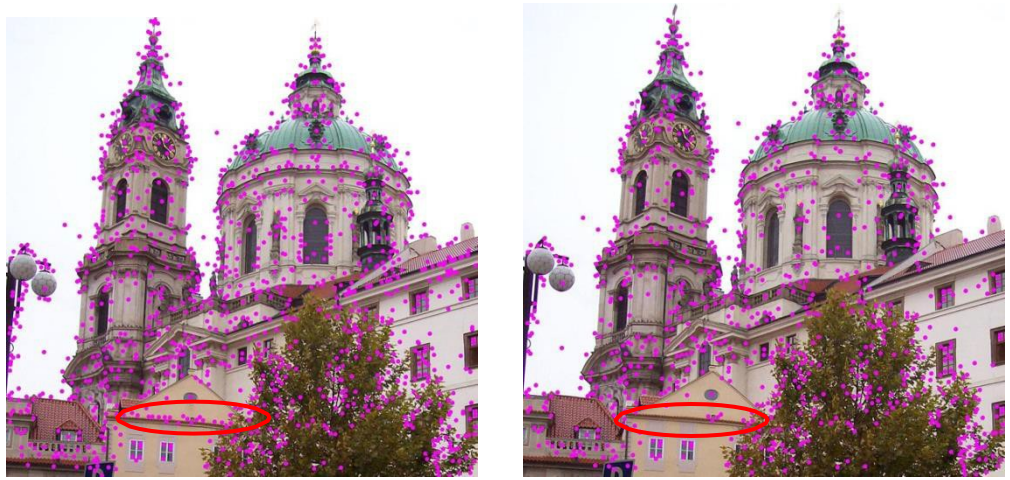
4. Non maximum suppression

- Threshold
  - Poorly localized candidates along an edge
- Principal curvature
  - DoG has strong response along edge
  - Compute ***Hessian matrix*** of D
  - $H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$

$$Tr(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$
$$Det(H) = D_{xx}D_{yy} + (D^{xy})^2 = \lambda_1\lambda_2$$

  - Remove outliners by evaluating
  - $\frac{Tr(H)^2}{Det(H)} = \frac{(\lambda_1+\lambda_2)^2}{\lambda_1\lambda_2} = \frac{(r+1)^2}{r}$ where $r = \frac{\lambda_1}{\lambda_2}$
  - if $r > k$ remove it (e.g. $r > 10$)



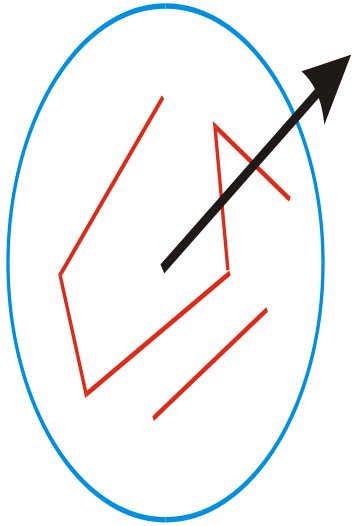


SIFT algorithm filters out those located on edges

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

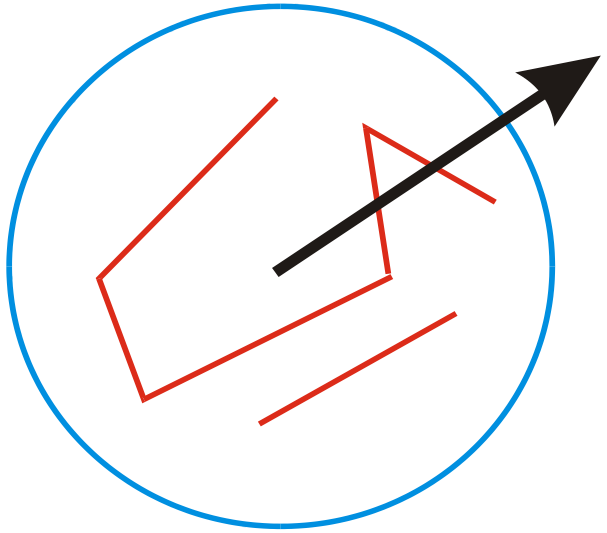
5. Becoming Rotation Invariant

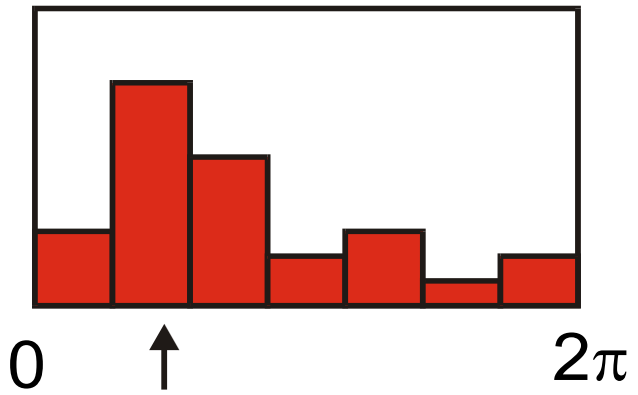
- Given a keypoint and its scale from DoG
- Select a ***characteristic orientation*** for the keypoint (based on the most prominent gradient there)
- Describe all feature relative to this orientation
- Causes features to be rotation invariant
  - If the keypoint appear rotated in another image, the features will be the same because they are relative to the characteristic orientation



5. Becoming rotation invariant

- Choosing characteristic orientation:
  - Use the blurred image associated with the keypoint's scale. Look at pixels in a square around it (say, size 16x16)
  - Compute the gradient direction of each pixel(by vertical and horizontal edge filters)
  - Create a histogram of these local gradient directions
  - Keypoint orientation = the peak of the histogram
  - Minor details: we'll weight each pixel's histogram contribution by the magnitude of its gradient and how close it is to the keypoint
  - Each key specifies stable 2D coordinates (x, y, scale, orientation)





Gradient magnitude and orientation

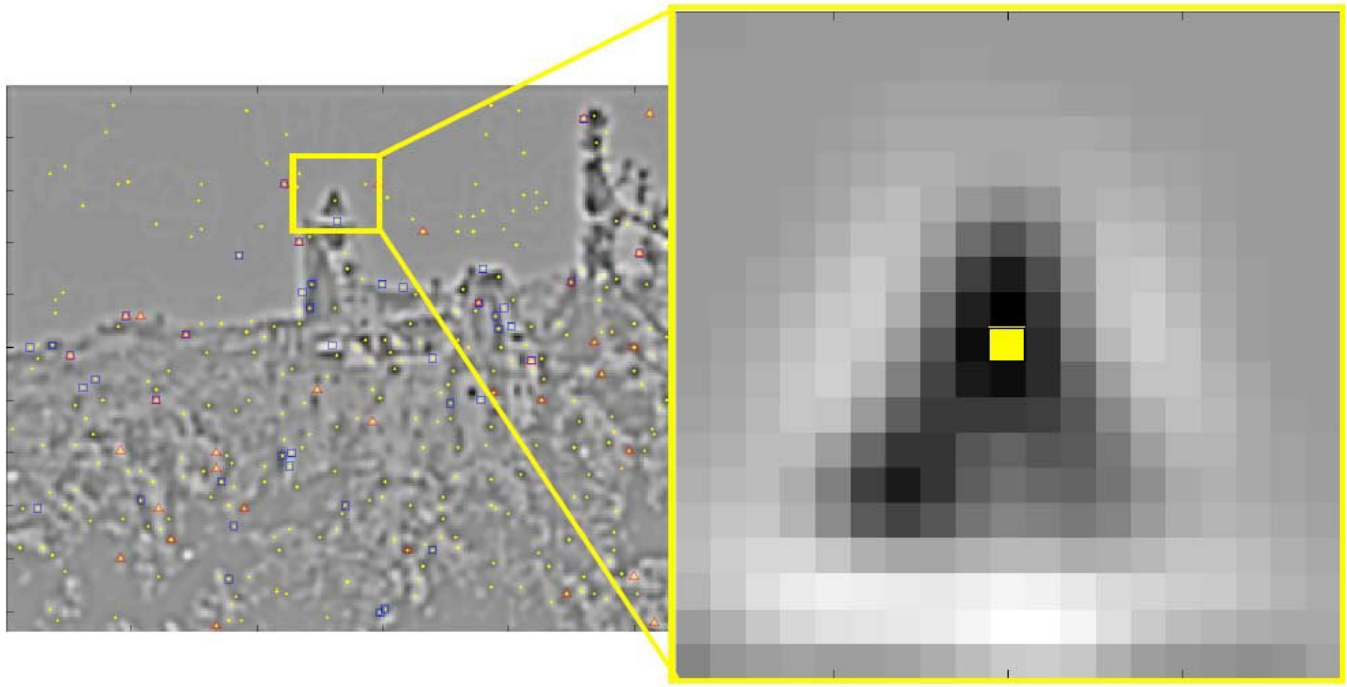
- At the scale of key point (x,y)
- Gradient is given

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

- Direction

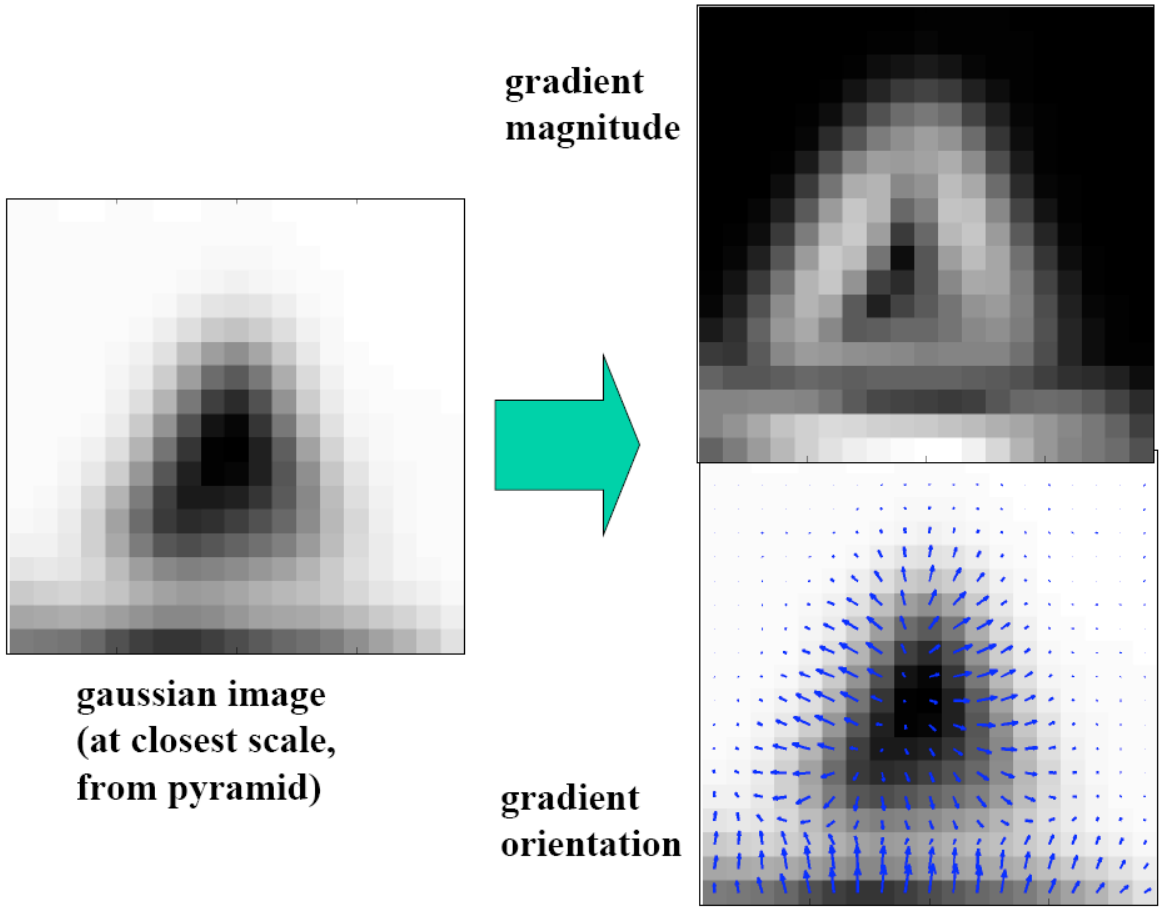
$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

Example: SIFT Orientation Assignment



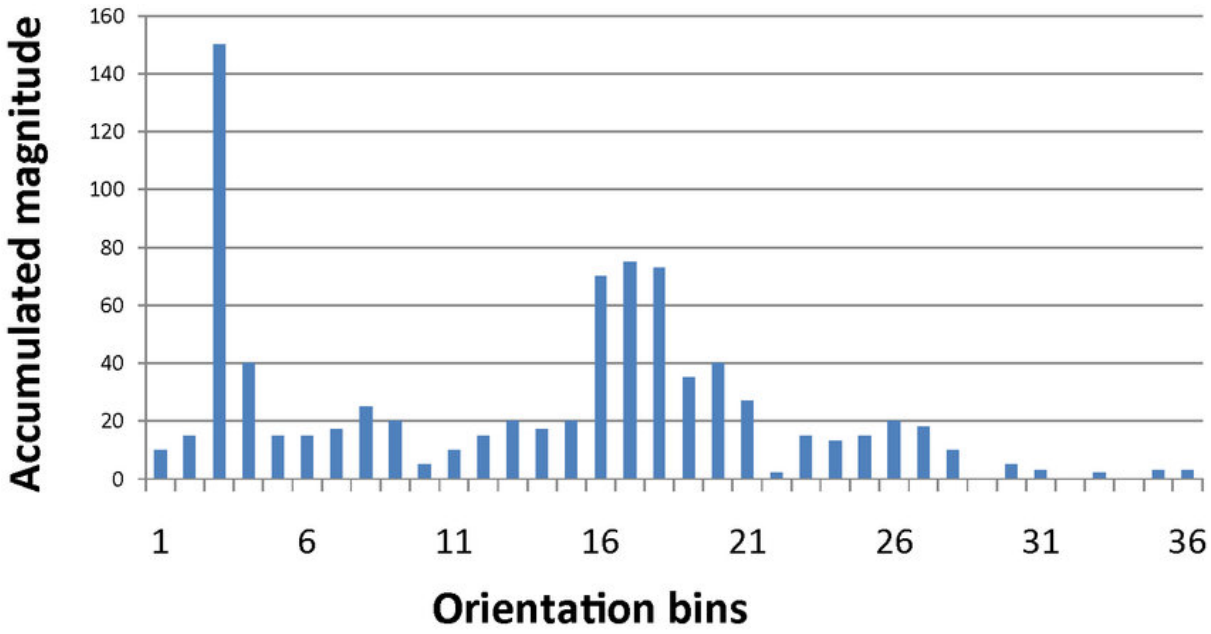
•Keypoint location = extrema location
•Keypoint scale is scale of the DOG image

Example: SIFT Orientation Assignment

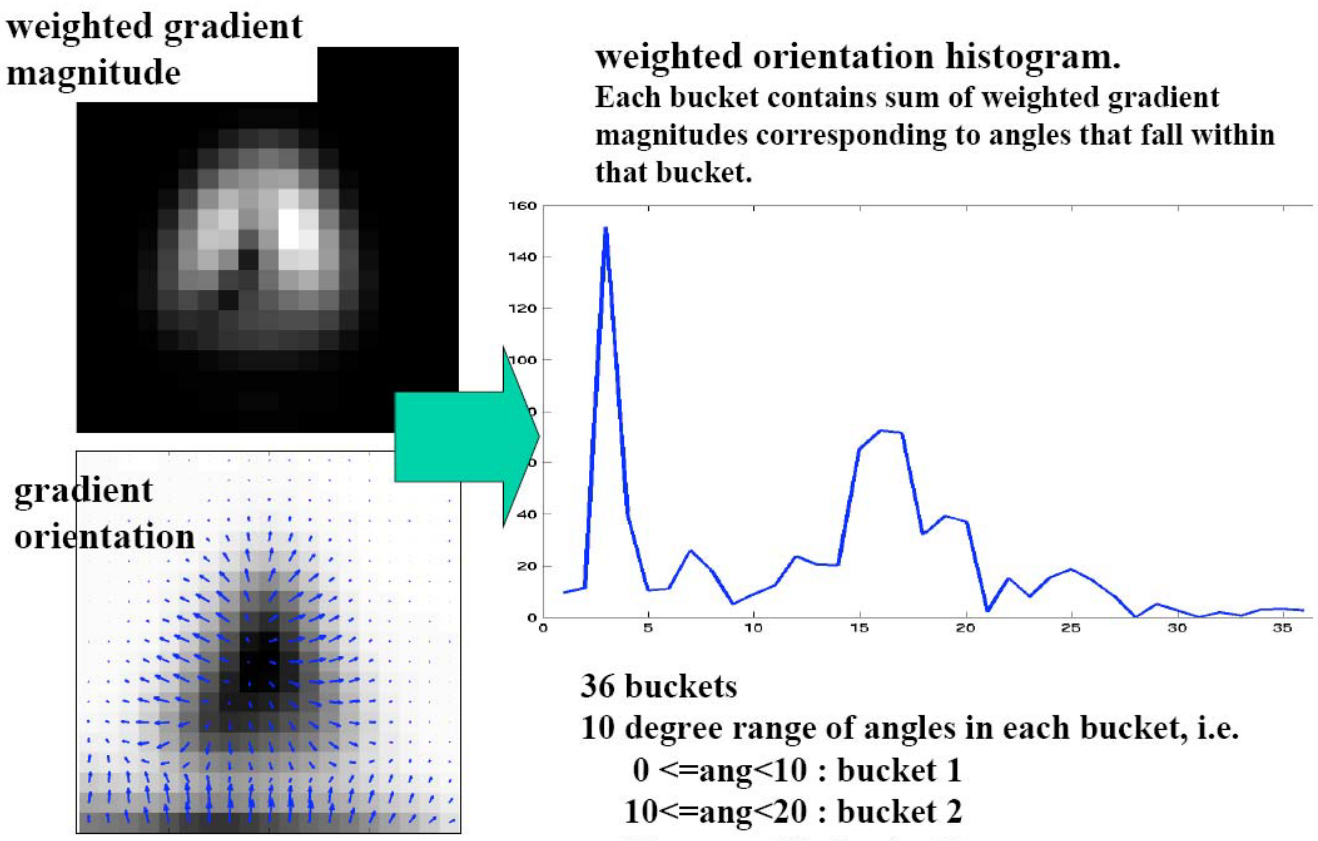


Creating a histogram of local gradient

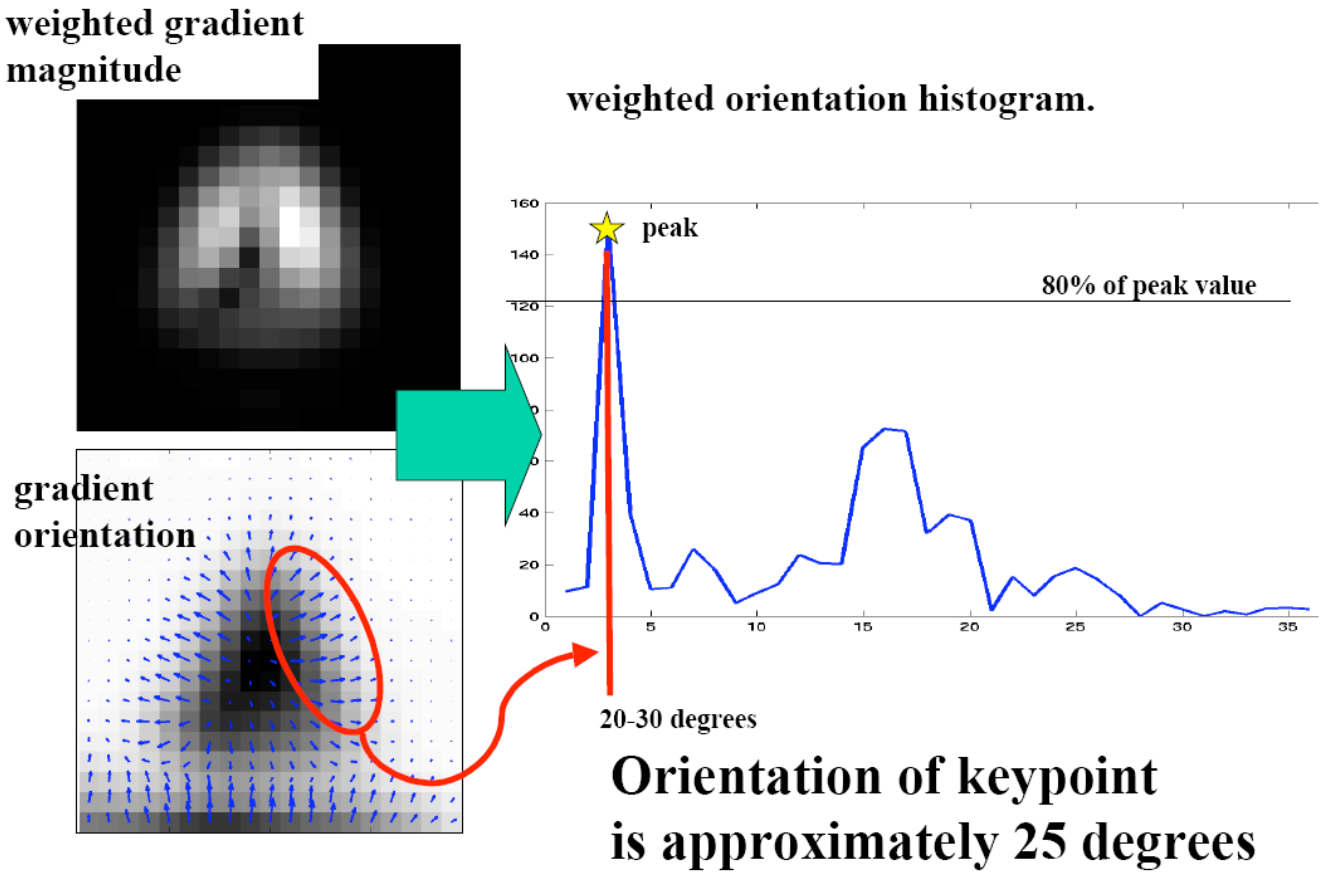
1. Compute orientation of a 36 bin histogram for each point
2. Select the bin with most orientation as the dominant orientation
3. Normalization: Rotate to the dominant orientation



Example: SIFT Orientation Assignment



Example: SIFT Orientation Assignment



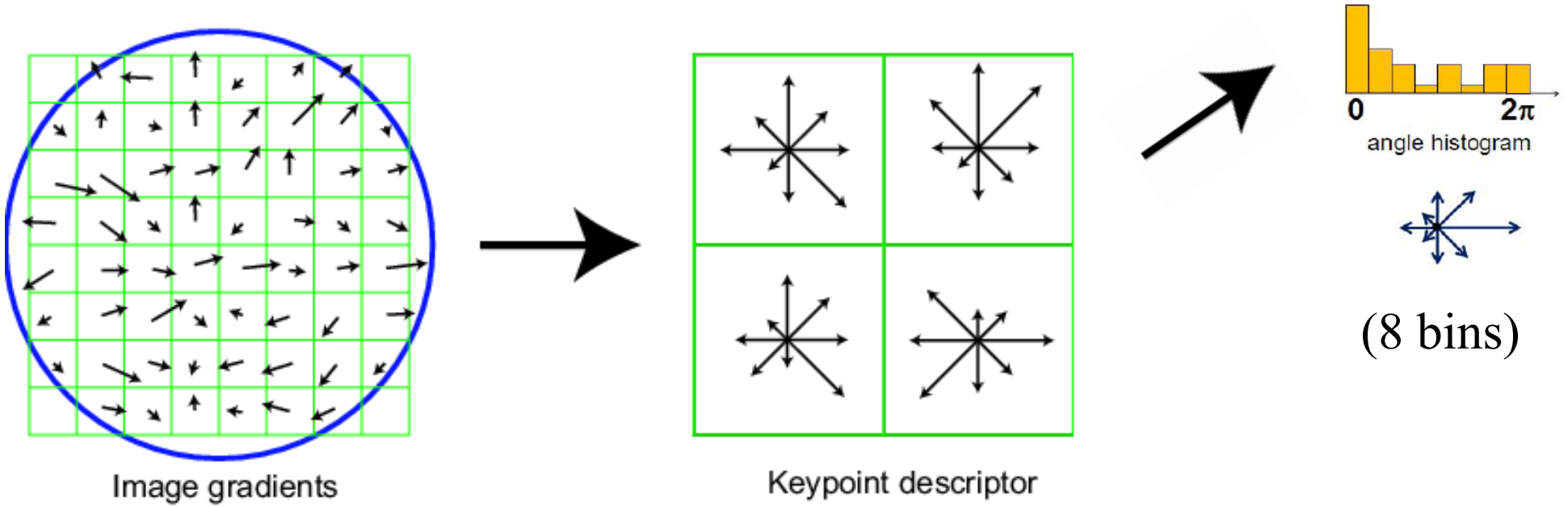
5. SIFT Orientation Assignment

- Select the peak as direction of the key point

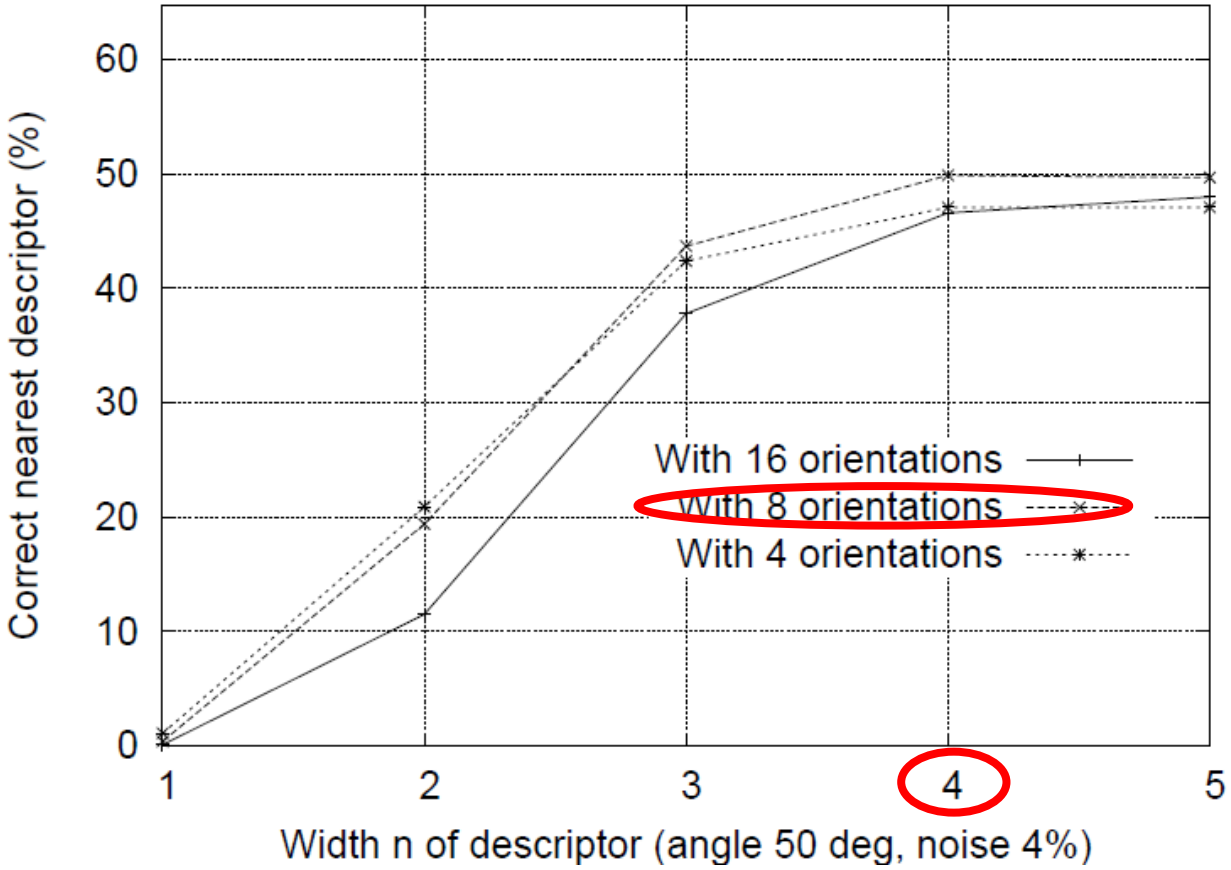
- Any peak within 80% of the highest peak is used to create another keypoint with that orientation at the same location

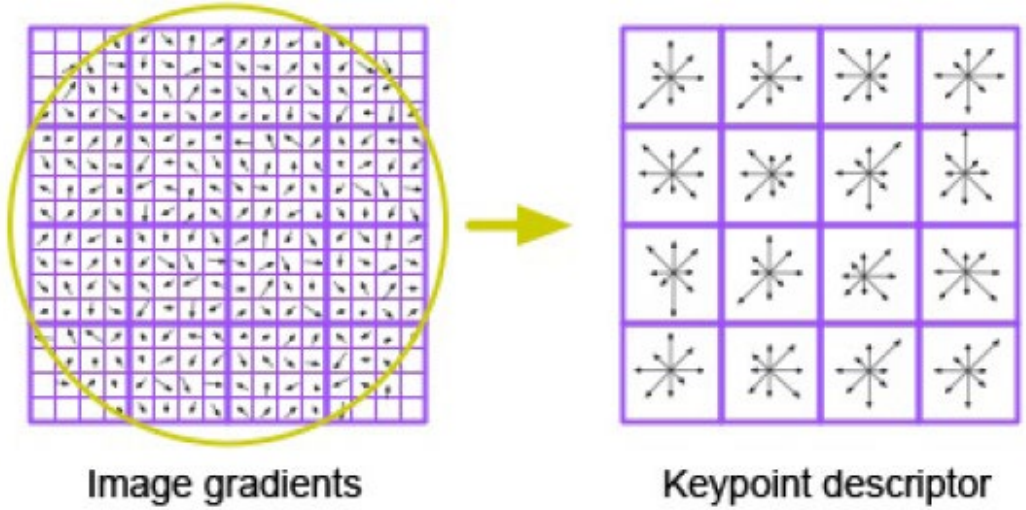
6. SIFT Descriptor formation

- Threshold image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms (8-bins)
- 8 orientations x 4x4 histogram array = 128 dimensions



Number of orientations (bin) in each histogram

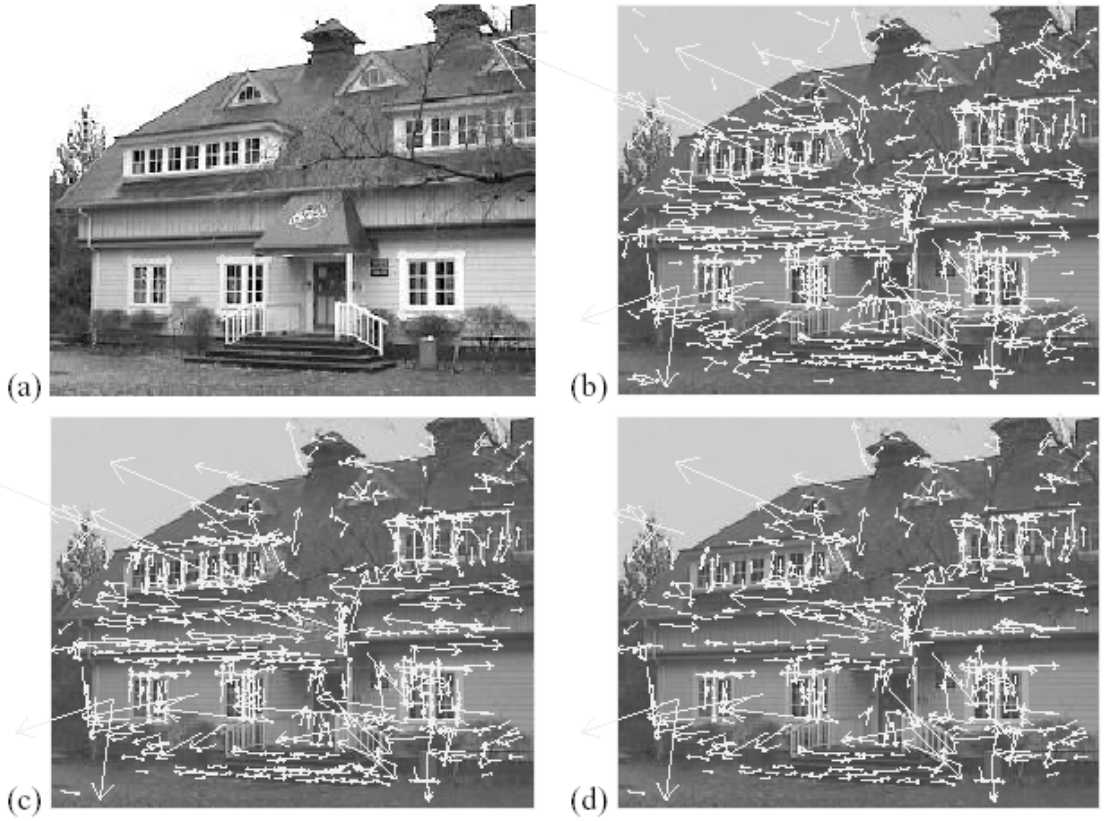




128 features

Image Credit: George Bebis

Example of key points detection



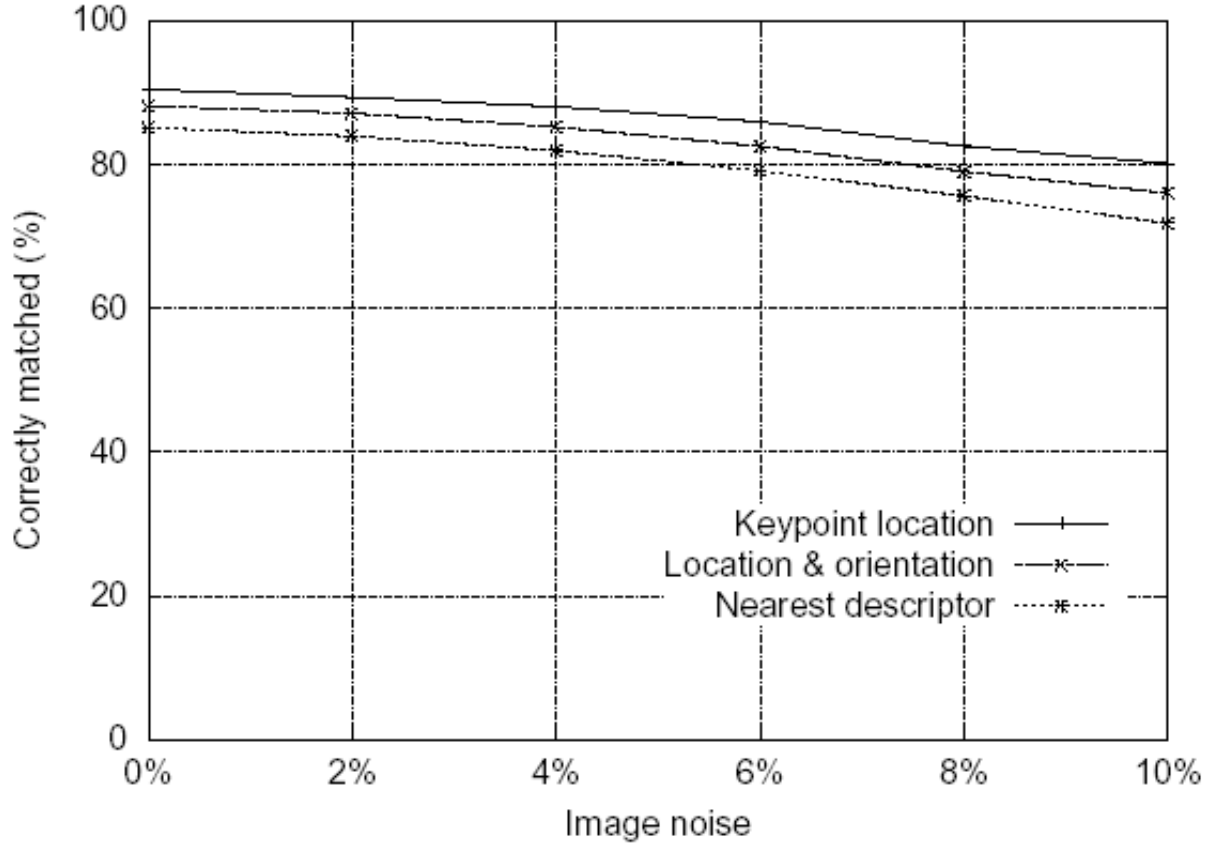
a) 233x189 image
b) 832 DoG extrema
c) 729 left after peak value threshold
d) 536 left after testing ratio of principle curvatures

Vectors indicate, scale, orientation and location.

Soure: David Lowe.

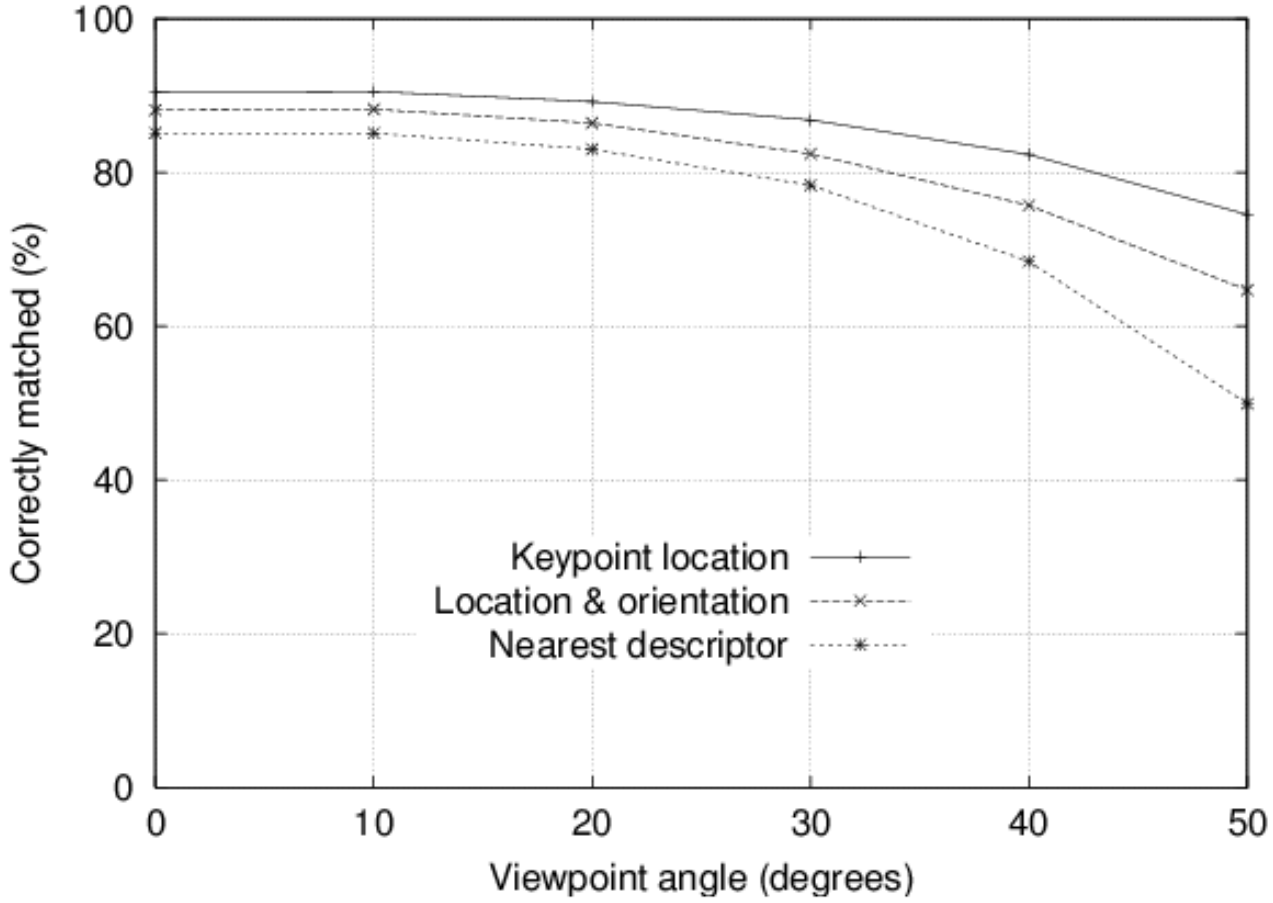
Feature stability to noise

- Match features after random change in image scale & orientation, with differing levels of image noise
- Find nearest neighbor in database of 30,000 features



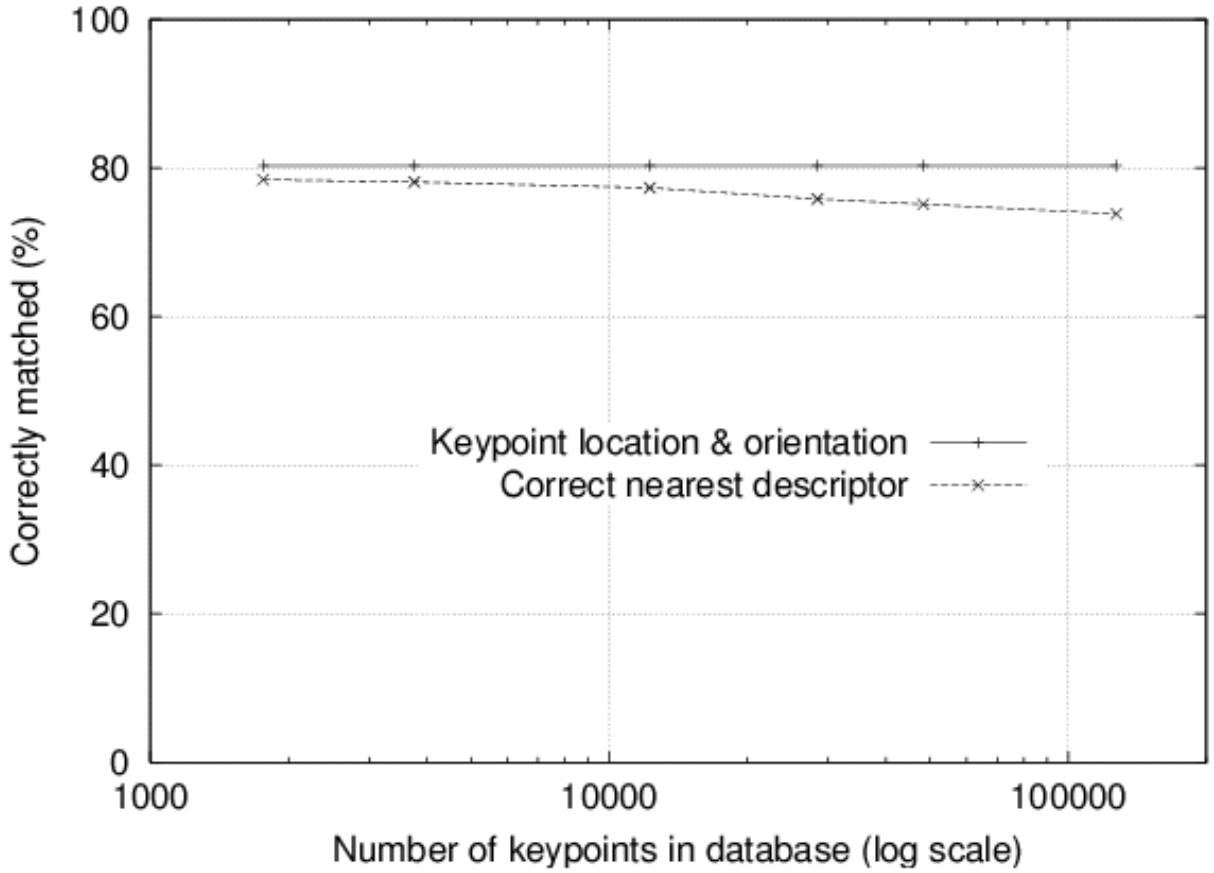
Feature stability to affine change

- Match features after random change in image scale & orientation, with 2% image noise, and affine distortion
- Find nearest neighbor in database of 30,000 features



Distinctiveness of features

- Vary size of database of features, with 30 degree affine change, 2% image noise
- Measure % correct for single nearest neighbor match



Example

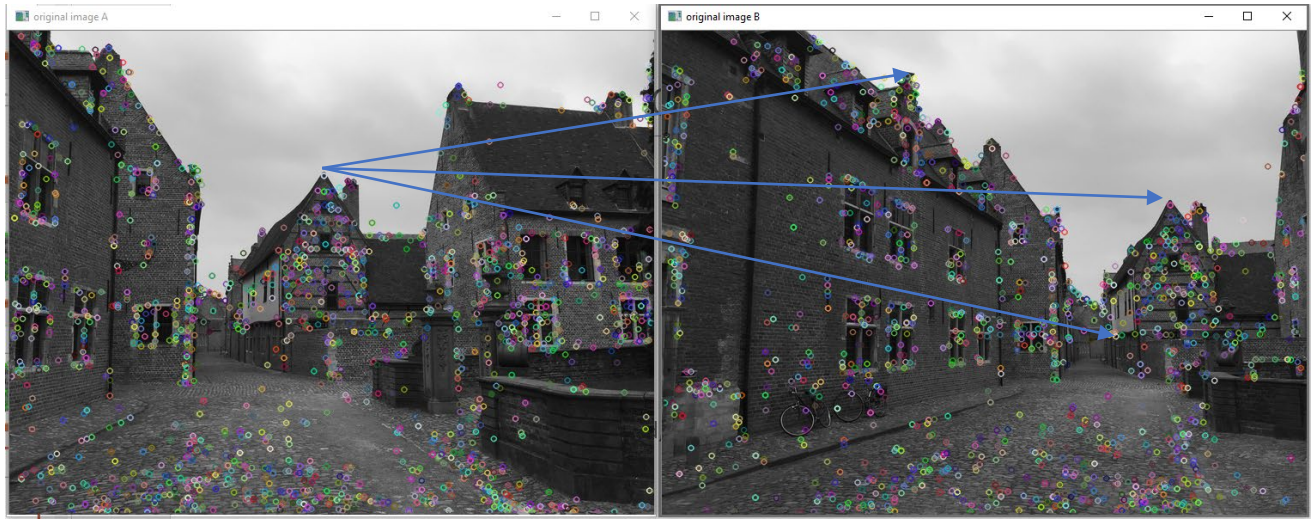
Image Source: www.vlfeat.org

Output of SIFT Descriptor

- One image yields
  - n 128-dimension descriptors: each one is a histogram of the gradient orientations with 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 recording the position of the patches [n x 2 matrix]

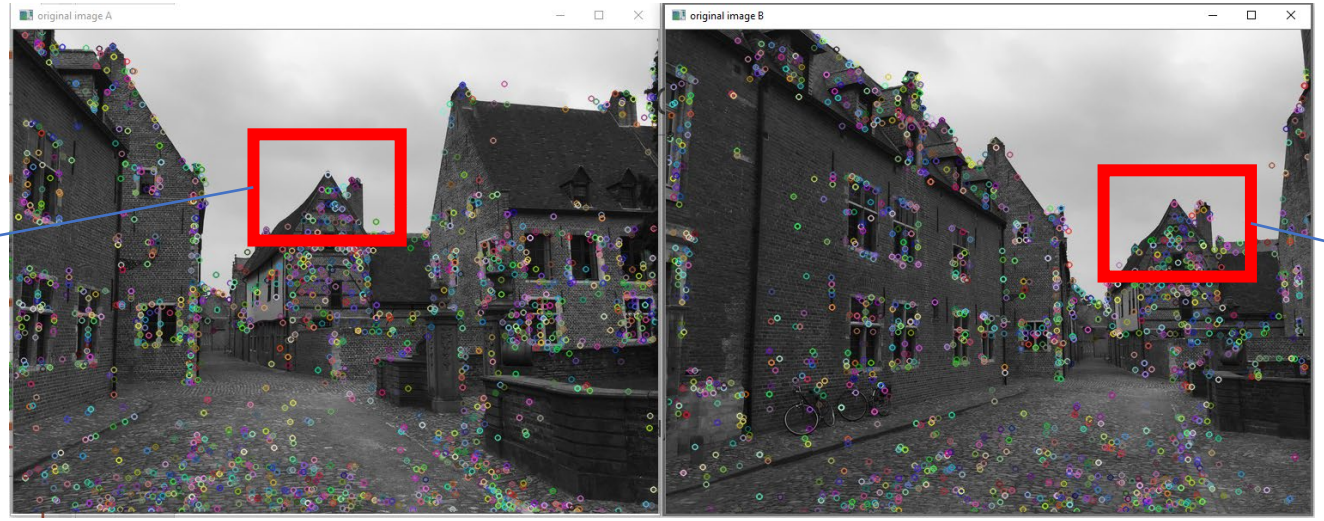
What next?

Matching the local features



- Once the SIFT descriptor is generated, we can use the descriptor for matching purpose.
- How?

Matching descriptors



$$\begin{pmatrix} x \\ y \\ \sigma \\ \theta \\ D1_{p1} \\ D2_{p1} \\ \vdots \\ D128_{p1} \end{pmatrix}$$

Compare

$$\begin{pmatrix} x \\ y \\ \sigma \\ \theta \\ D1_{p2} \\ D2_{p2} \\ \vdots \\ D128_{p2} \end{pmatrix}$$

How to measure the distance?

$$p1 = \begin{pmatrix} x \\ y \\ \sigma \\ \theta \\ D1_{p1} \\ D2_{p1} \\ \vdots \\ D128_{p1} \end{pmatrix} \quad \text{Distance?} \quad p2 = \begin{pmatrix} x \\ y \\ \sigma \\ \theta \\ D1_{p2} \\ D2_{p2} \\ \vdots \\ D128_{p2} \end{pmatrix}$$

Distance Measurement Method 1:

Using SSD Euclidean Distance

$$SSD(p1, p2) = \sqrt{\left(D1_{p1} - D1_{p2}\right)^2 + \left(D2_{p1} - D2_{p2}\right)^2 + \ \ldots + \left(Dn_{p1} - Dn_{p2}\right)^2}$$

How to measure the distance?

$$p1 = \begin{pmatrix} x \\ y \\ \sigma \\ \theta \\ D1_{p1} \\ D2_{p1} \\ \vdots \\ D128_{p1} \end{pmatrix} \quad \text{Distance?} \quad p2 = \begin{pmatrix} x \\ y \\ \sigma \\ \theta \\ D1_{p2} \\ D2_{p2} \\ \vdots \\ D128_{p2} \end{pmatrix}$$

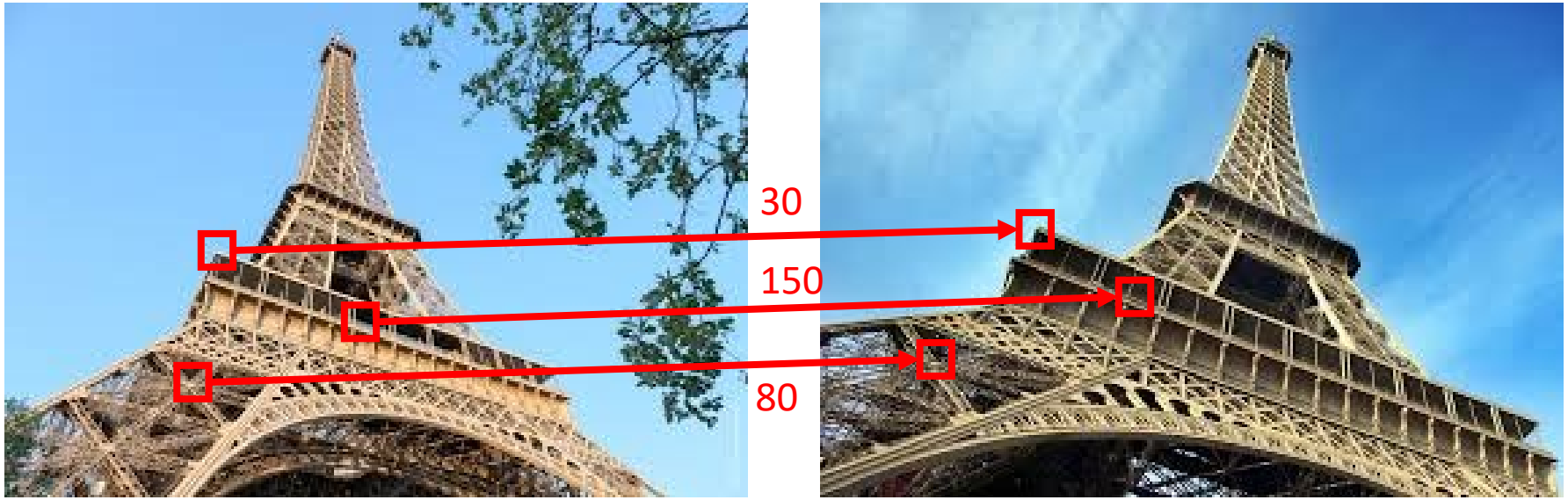
Distance Measurement Method 2:

Using Cosine Distance

$$d(p1, p2) = \frac{p1 \cdot p2}{\|p1\|\|p2\|}$$

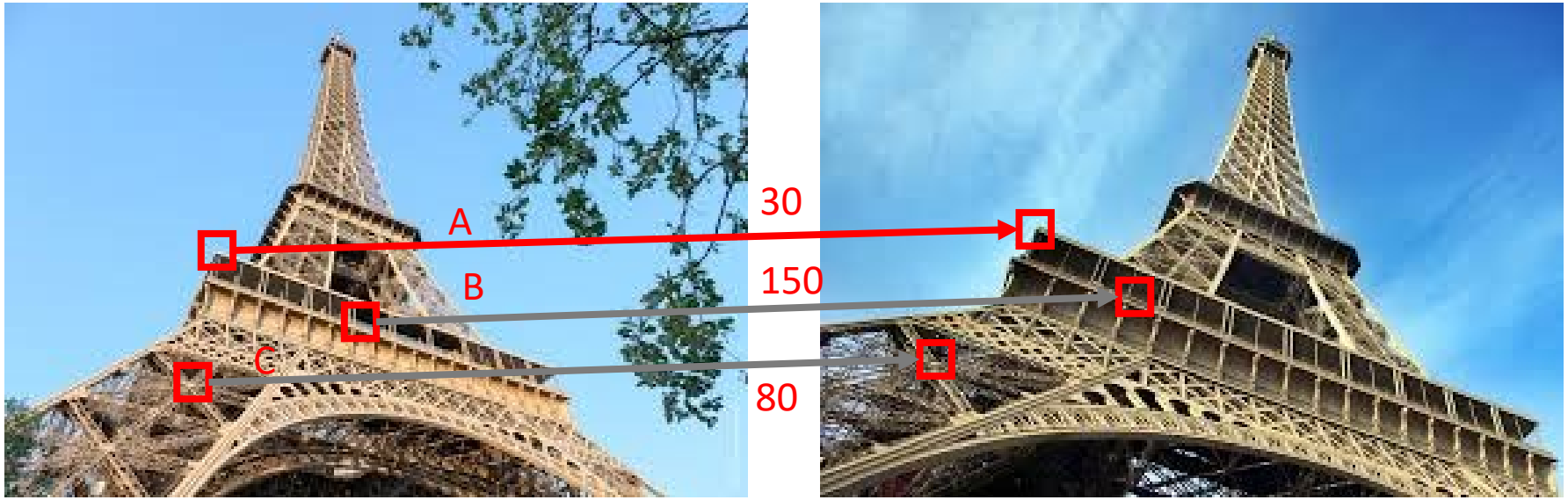
- Either ***SSD*** or ***cosine distance*** can be adopted.
- Pairing Method: The vector with the smallest distance are parried.
- Is this the best way to do?

Define a threshold value?



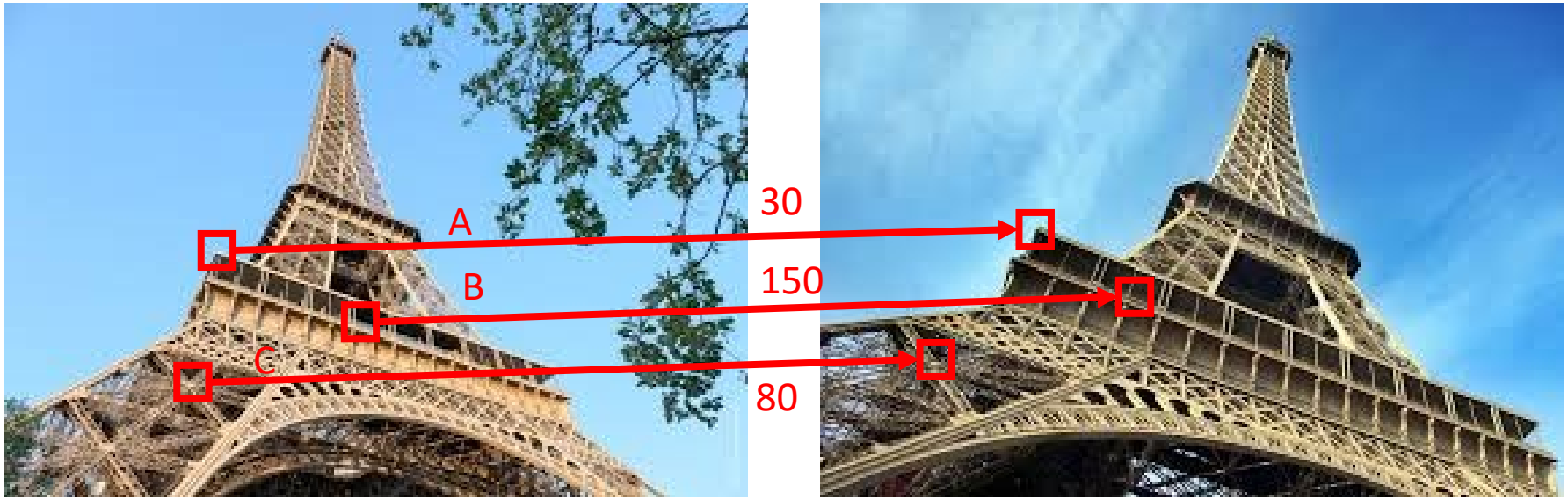
- Could we use ***small threshold***?
- Rule: Accept match if $ssd(p1, p2) < T$ where $T = 50$
- What is the effect?

Choosing a Small Threshold



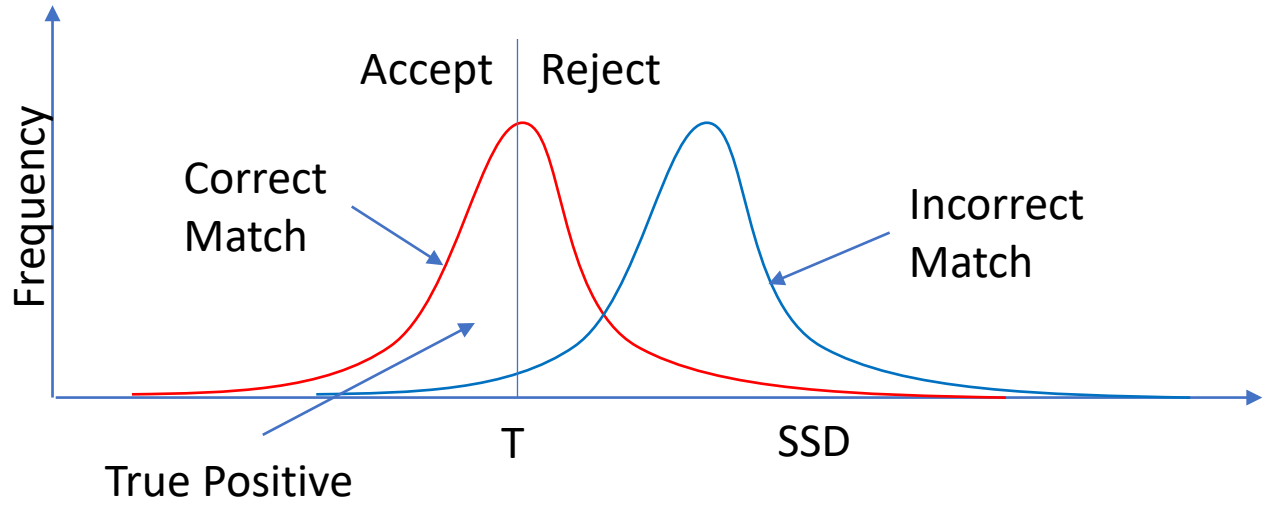
- When $T = 50$
- Only A is matched which is known as “true positive”
- B and C are rejected.

Choosing a Large threshold



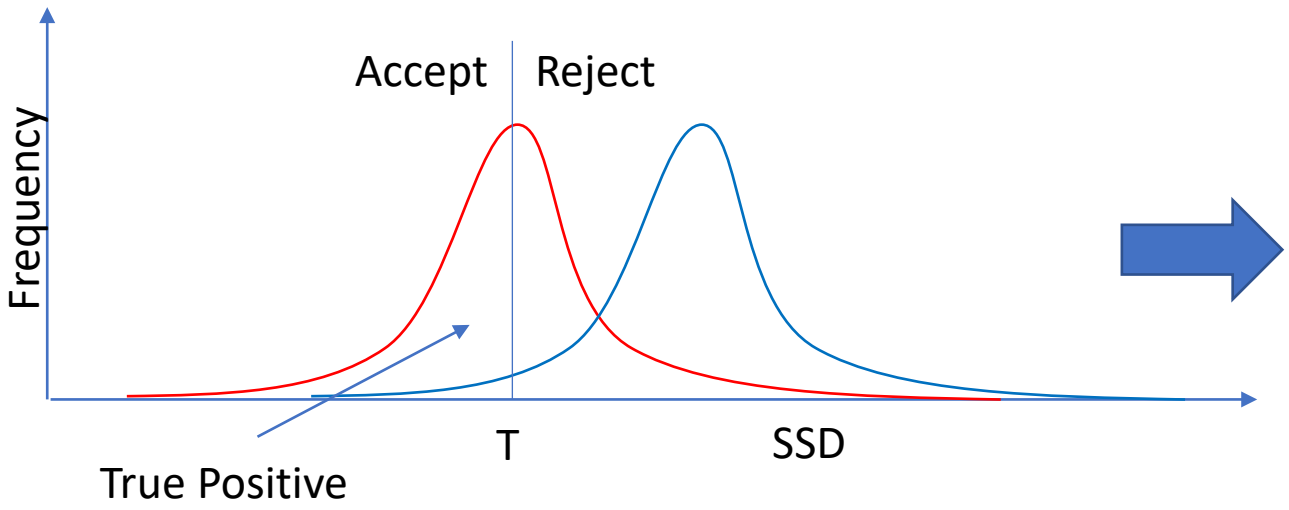
- Rule: Accept match if $\mathrm{SS}d(p1, p2) < T$ where $T = 200$
- What is the effect?
  - A, B and C are all accepted
  - A and B are “true positive”
  - C is “false positive”

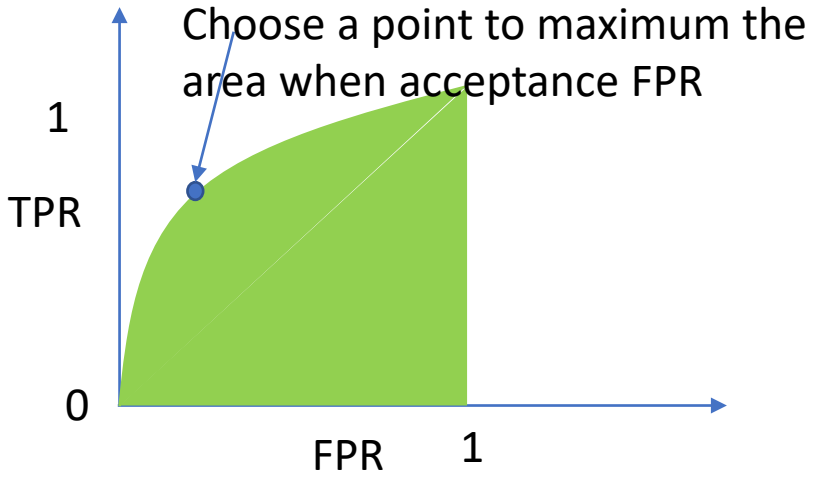
The effect of threshold value



- True Postive Rate $TPR = \frac{Num\ of\ true\ positive}{Num\ of\ actual\ matches}$
- False Postive Rate $FPR = \frac{Num\ of\ false\ positive}{Num\ of\ actual\ matches}$
- When T increase TPR increase but FPR also increase.
- How to choose the value of T?

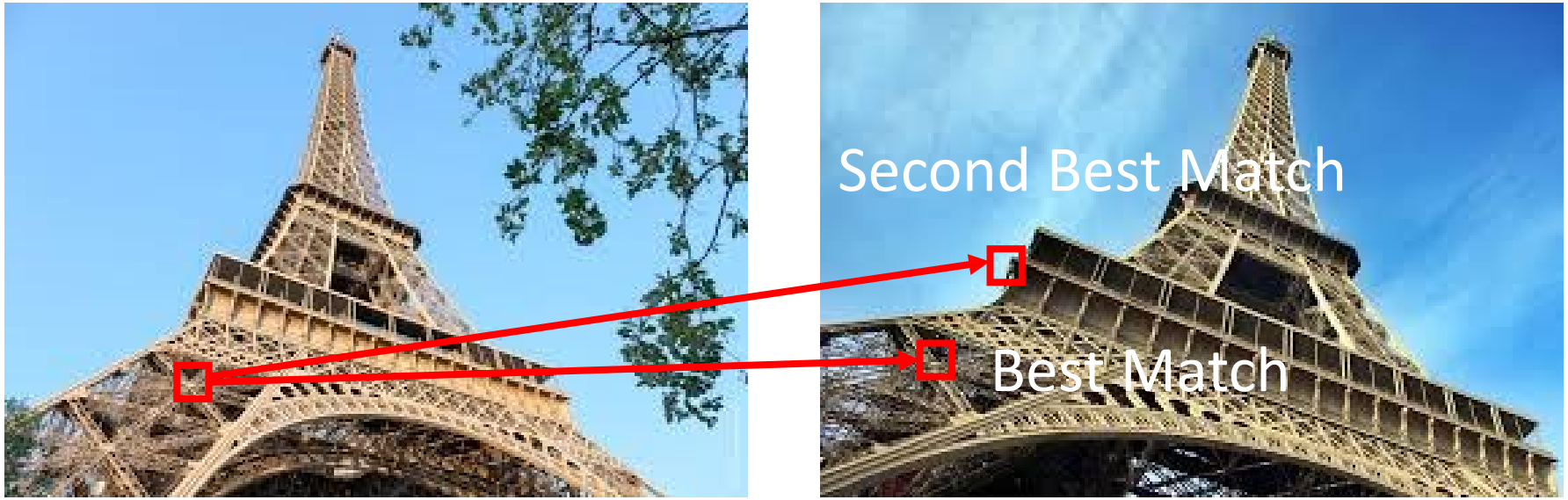
Receiver Operating Characteristic (ROC) Curve





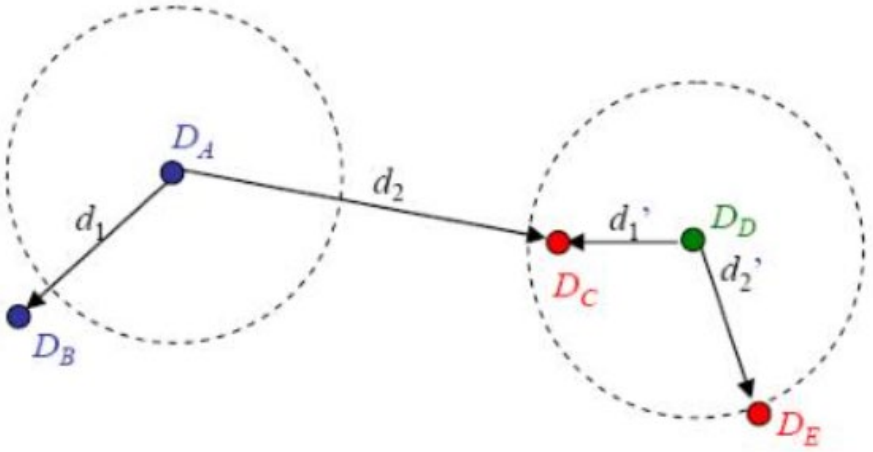
- More Information: [Receiver operating characteristic - Wikipedia]

Another Method-Near Neighbour Distance Ratio



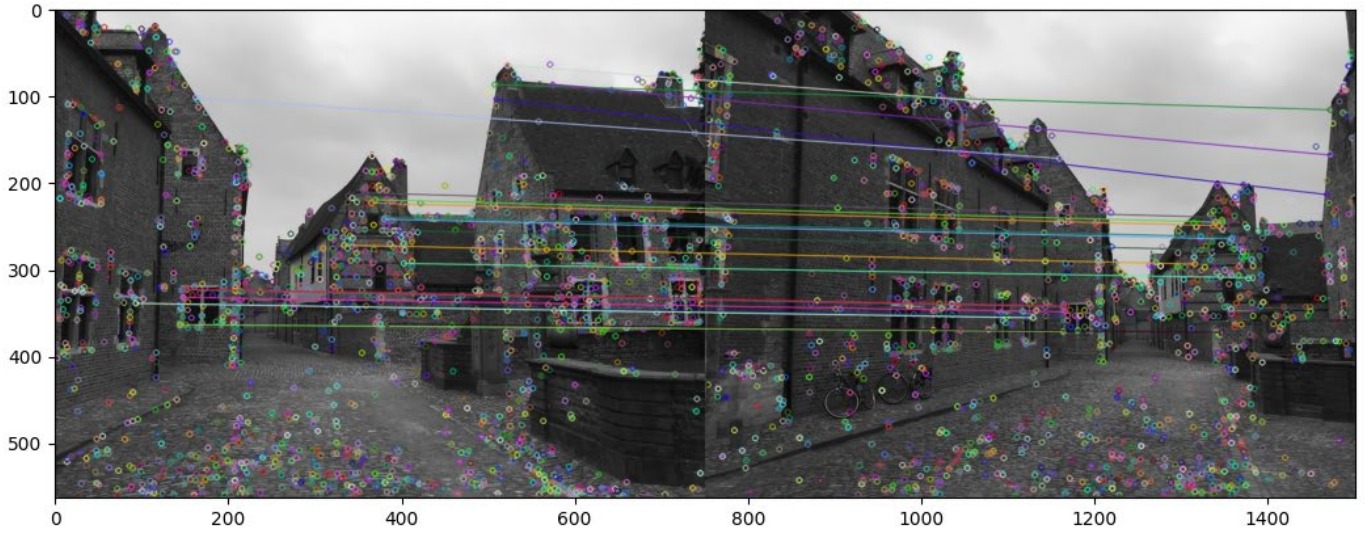
- Near neighbour distance ratio (NNDR)
- $NNDR = \frac{distance\ of\ Best\ Match}{distance\ of\ Second\ Match}$
- Accept matches if NNDR<Threshold

Example



- $NNDR(D_A) = \frac{d_1}{d_2} < \frac{1}{2}$ therefore match between D_A and D_B are accepted
- $NNDR(D_D) = \frac{d_c}{d_E} > \frac{1}{2}$ therefore match between D_D and D_c are rejected

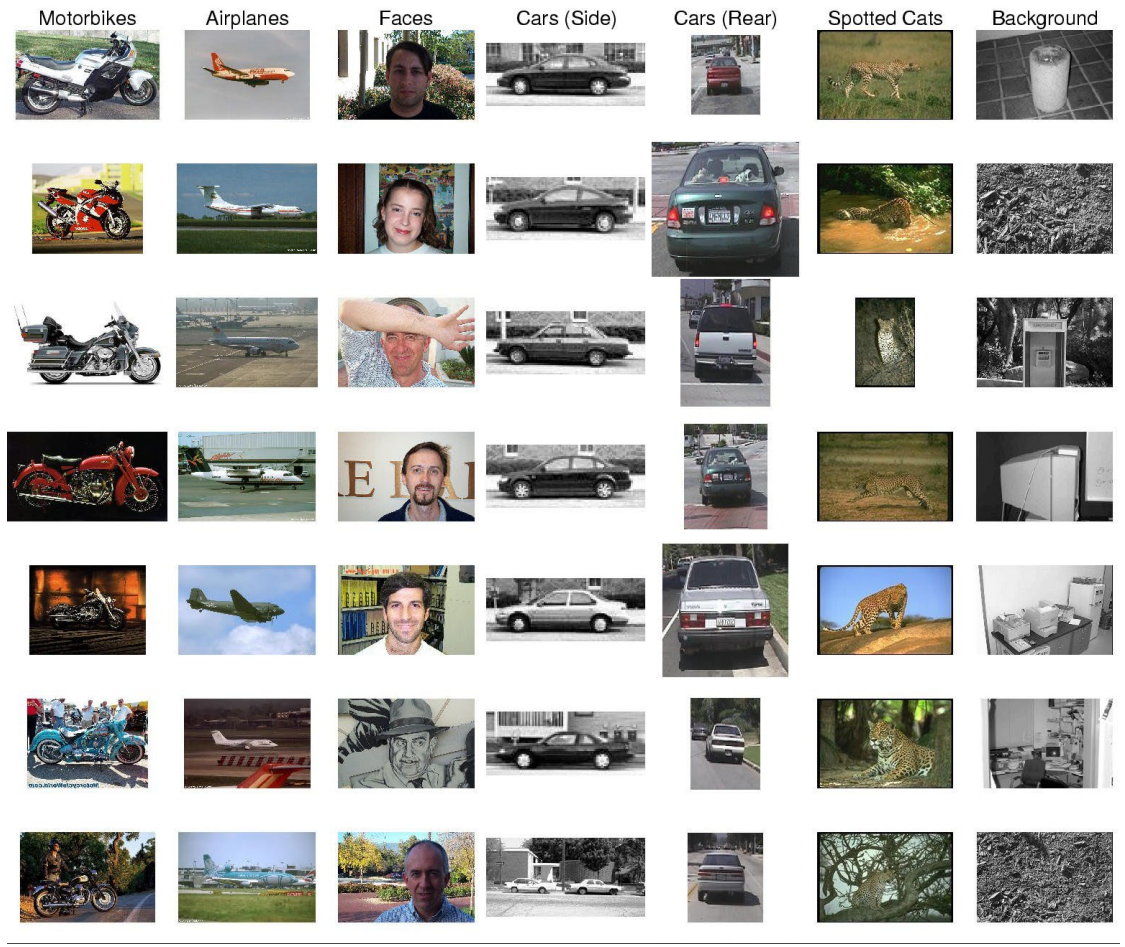
Match Result



Summarizes the steps for building SIFT Descriptor

1. *Constructing a scale space* - Building the smoothing versions of the image with different σ value and image sizes (octaves).
2. *LoG Approximation by DoG* – Approximate LoG by DoG which is less computational expensive.
3. *Finding key points* - Search for maxima/minima in DoG images across scales
4. *No maxima suppression* – Eliminate the low contrast key points and edges
5. *Assigning an orientation* – The orientation of each pixel is calculated, and canonical orientation is defined at the peak. All pixels' orientation are defined based on the canonical orientation. The orientation is quantiles into 8 orientations(bins) and histogram is constructed.
6. *Generate SIFT feature description* – generate the 8-bin histogram of 4x4 regions making *128 dimensions descriptor*.

Other Applications – Image Classifications



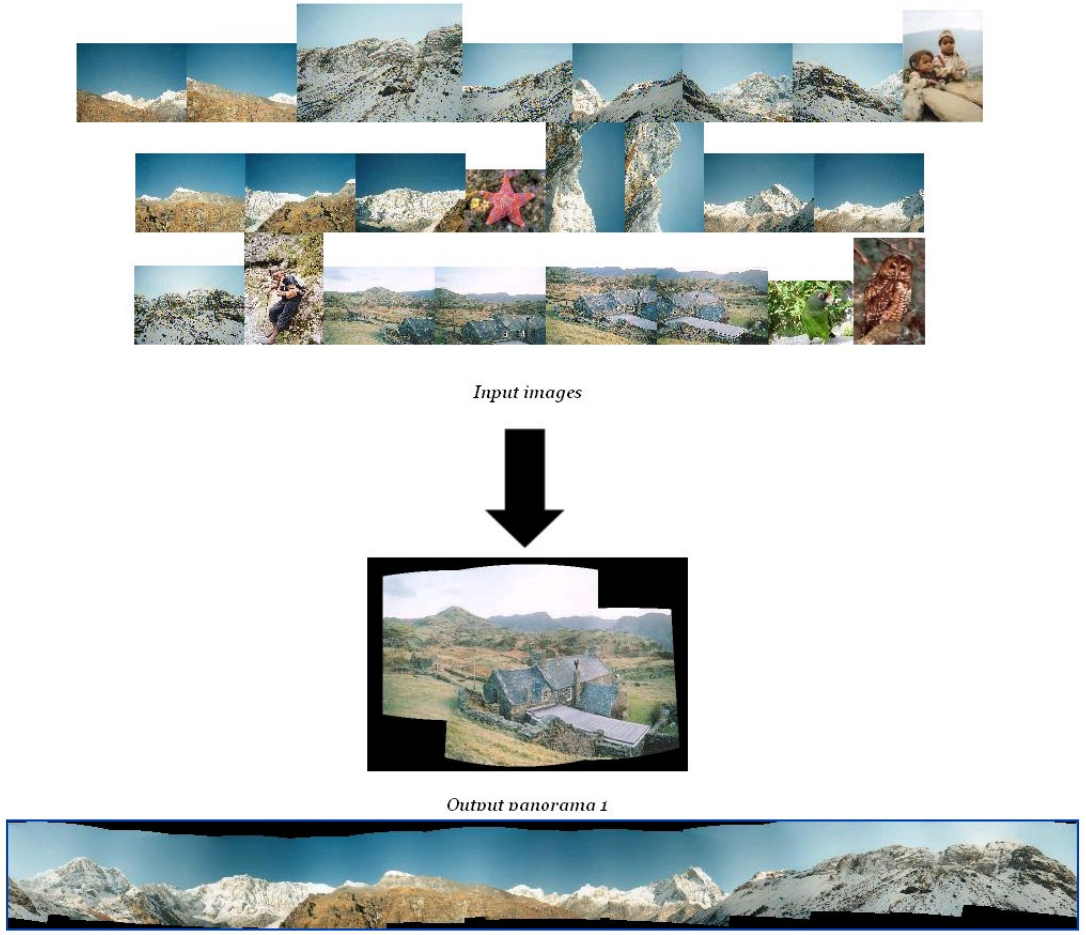
Slide Credit: George Bebis

Image Retrieval

change in viewing angle

Slide Credit: George Bebis

Image panoramas from an unordered image set



Slide Credit: George Bebis

Today's Agenda

- Shape Descriptor
  - Scale Invariant Feature Transform (SIFT)
  - Speed Up Robust Feature (SURF)
  - Oriented FAST and Rotated BRIEF (ORB)
  - Histogram of Oriented Graph (HOG)

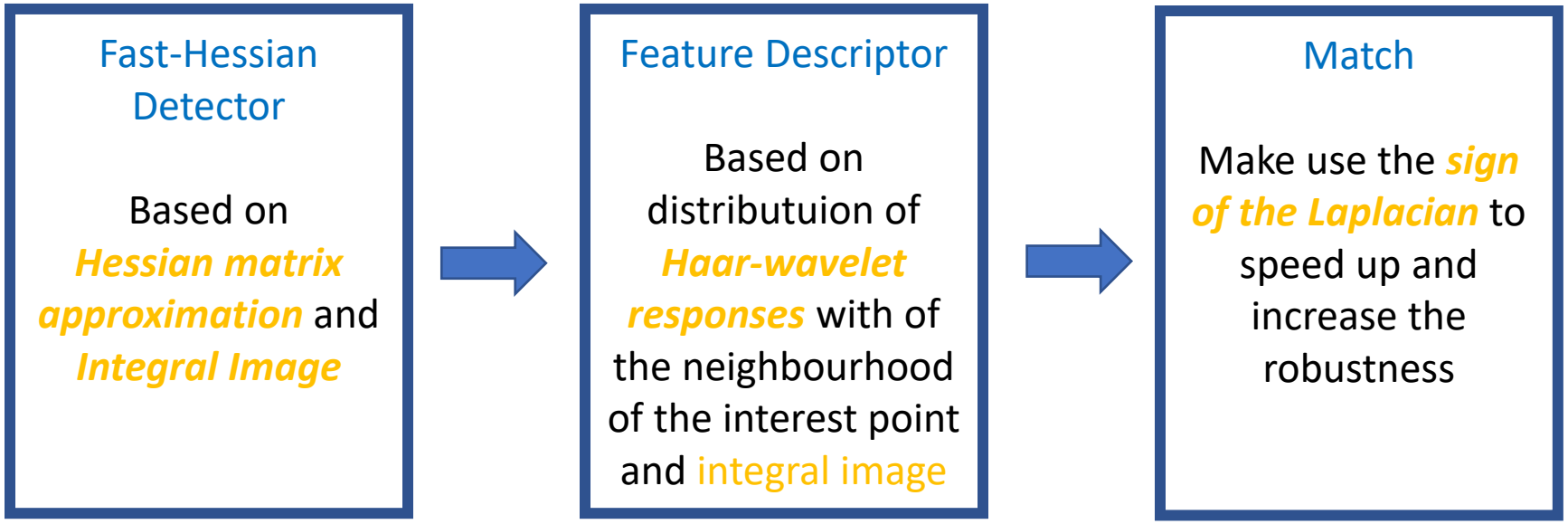
- Visual Bags of Words.

Speeded Up Robust Features (SURF)

- Issue of SIFT:
  - SIFT descriptor is robust. However the high dimension (128-dimensions) of the descriptor make the extraction and comparison time consuming.
- Motivation of SURF:
  - To develop a detector and descriptor which similar performance of SIFT but run faster.
- Method:
  - Fast interest point detector
  - Faster descriptor matching
  - Invariant to common image transformation
    - Rotation invariant
    - Scale invariant
    - Illumination change
    - Small view point change

Speeded Up Robust Features (SURF)

- Three Steps in SURF



Fast Hessian Detector for Feature Extraction

- Surf detector uses ***Hessian matrix*** for finding maxima or minima because of its good performance in computation time and accuracy.

- The Hessian matrix applied on image each pixel is:

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

SURF – Hessian matrix at different scale

- SURF detector relies on the determinant of the Hessian for location at scale changes. The image is filtered by a Gaussian kernel.

- The Hessian matrix

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

Where

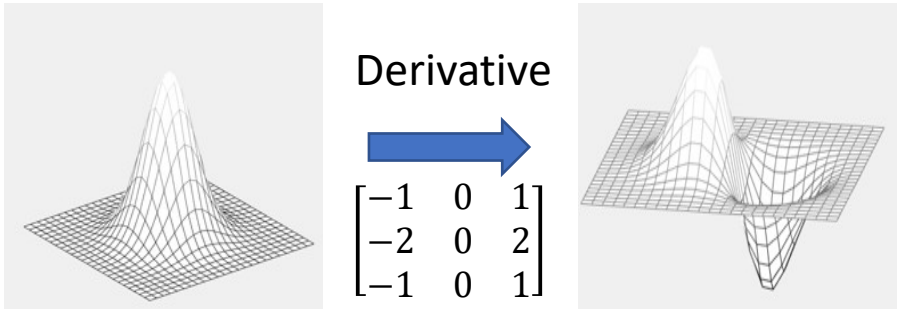
$L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the LOG or Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$

$L_{yy}(\mathbf{x}, \sigma)$ is the convolution of the LOG or Gaussian second order derivative $\frac{\partial^2}{\partial y^2} g(\sigma)$

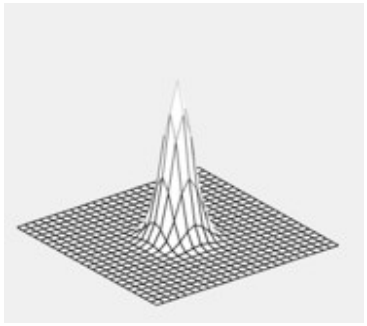
$L_{xy}(\mathbf{x}, \sigma)$ is the convolution of the LOG or Gaussian second order derivative $\frac{\partial}{\partial x}\frac{\partial}{\partial y} g(\sigma)$

Issues with Gaussian filter

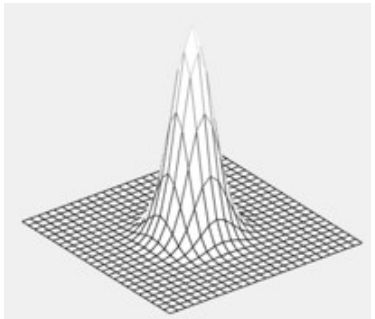
- Gaussian Filter is slow



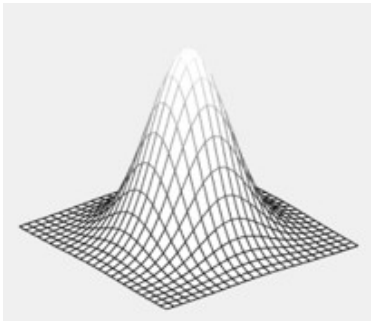
- Computational time increase as the filter size increase



$g(\sigma)$



$g(2\sigma)$



$g(3\sigma)$

Laplacian of Gaussian - Approximation

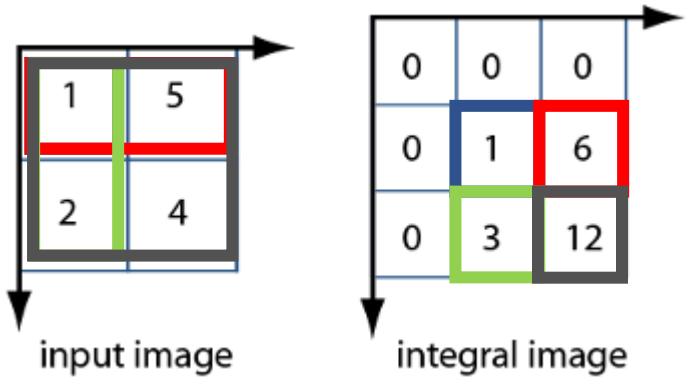
- In SIFT, the Laplacian of Gaussian is approximated by the DOG
- SURF uses similar approach to approximate the Laplacian of Gaussian by using a box filter with integral image.

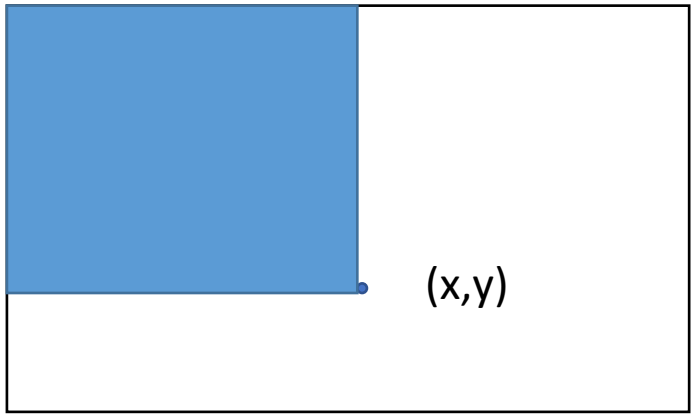
Integral Image

- In integral image, the value of a pixel is the sum of all pixel above and to the left

$$I_{\sum}(x,y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j)$$

Example:



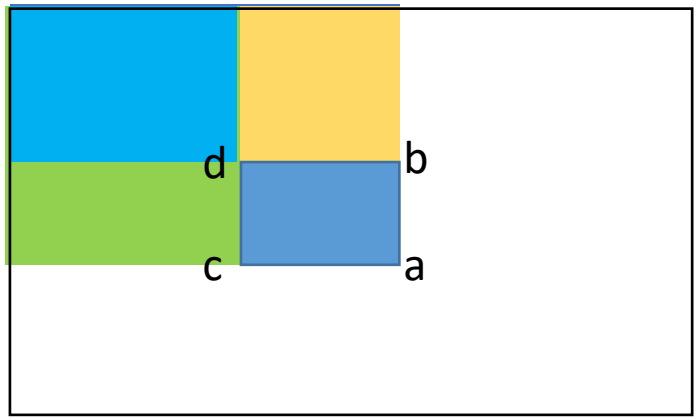


Efficient computation with integral image

- How to compute the sum of pixel inside the rectangular region using integral image?

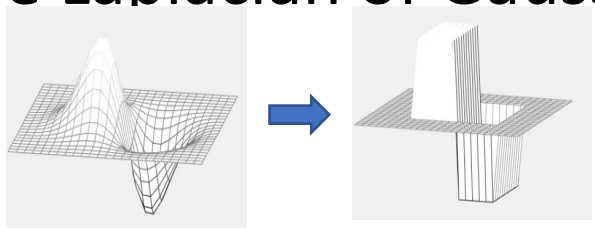
  $I_{abcd} = I_a - I_b - I_c + I_b$

- Only 3 operations compute the sum of pixel inside rectangular image regardless of the region sizes.

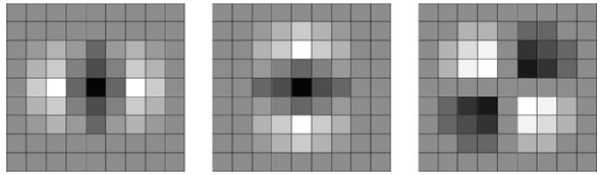


Approximation of Gaussian Filter

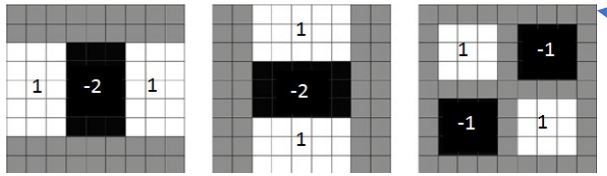
- The Laplacian of Gaussian filter is approximated by a box filter



- The Laplacian filters L_{xx}, L_{yy} and L_{xy}



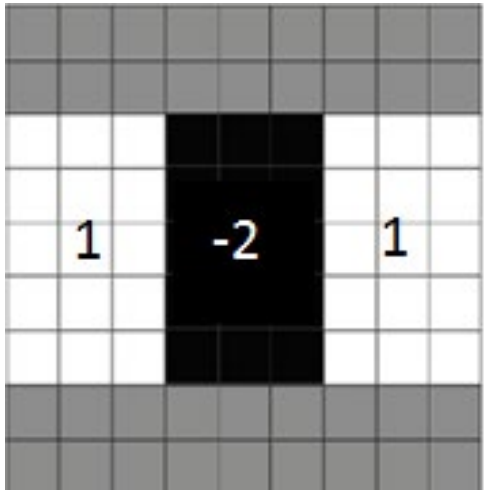
- The weight box filter approximation



Gray = 0
White = 1
Black = -1

Why is Integral image more efficient?

- Suppose we wish to calculate the response of an image with the L_{xx} box filter
- Without using integral image, it requires
  - 14 additions to add all white pixels in the left
  - 14 additions to add all black pixels in the middle
  - 1 multiplication (-2) to pixels in the middle
  - 14 additions to add all white pixels in the right
  - 2 additions to add all 3 region
  - Total: 14+14+14+2=44 additions and 1 multiplication
- When using integral image
  - 3 additions to calculate the overall region sum
  - 3 additions to calculate the middle black region
  - 1 multiplication to multiply -3 to sum of middle region
  - 1 subtraction to subtract the middle region from the total sum
  - Total = 3+3+1=7 additions and 1 multiplication



15 white x 2
15 black

Scale-space representation

- Scale spaces are usually implemented as image pyramids.
- Images are repeatedly smoothed with Gaussian and subsampled.
- SIFT adopted similar methods for feature detector across different scales.
- It is not computational efficient

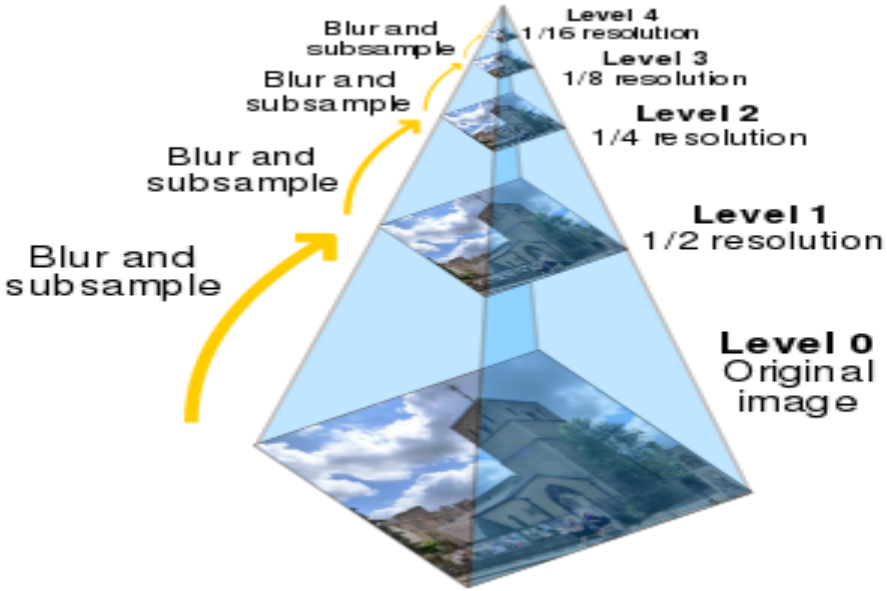
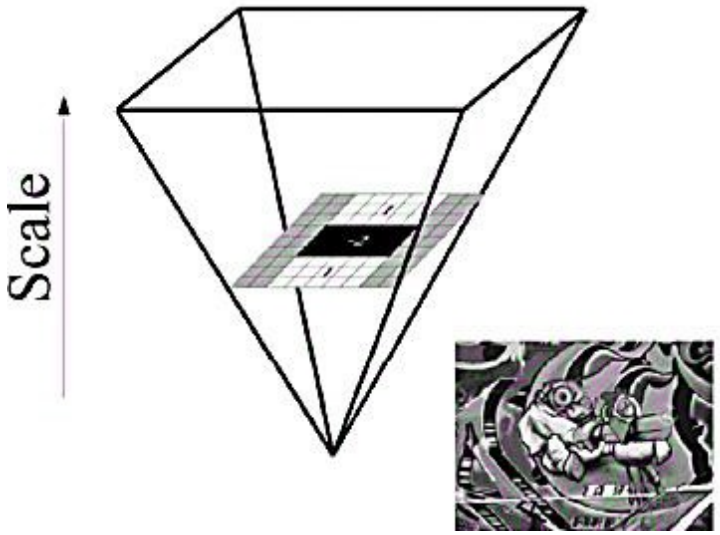


Photo credit: Wikipedia

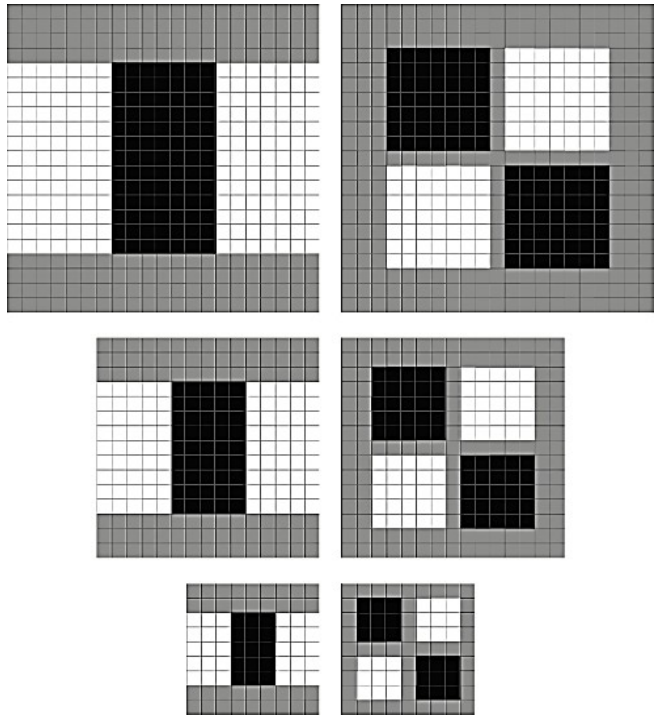
Filters with different sizes

- In SURF, box filters and integral image are adopted.
- The computation time is invariant to the filter size.
- The scale space can be created by repeatedly ***increasing the box filters size*** rather than down sampling the image

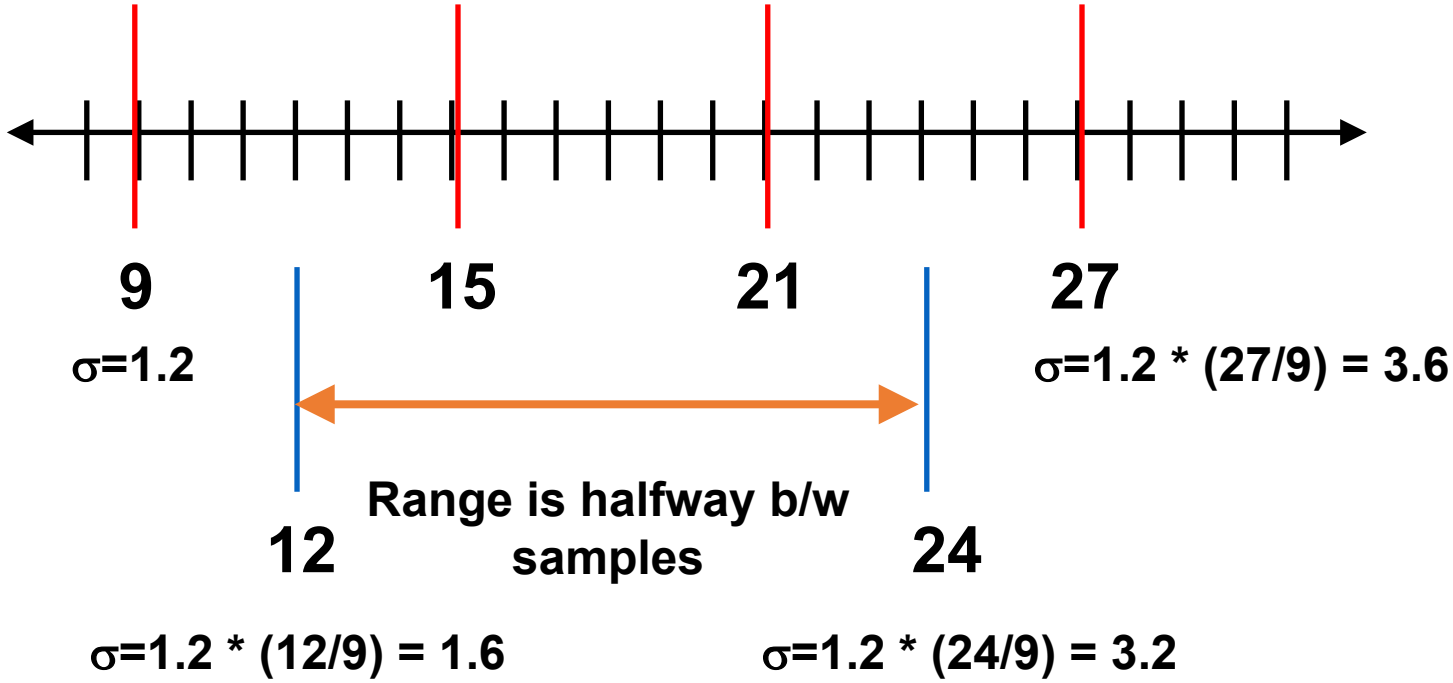


Filters with different sizes

- The scale-space is divided into a number of octaves referring to a series of responses across different scale
- In SURF, the size of box filters starts with 9x9 which correspond to σ=1.2
- Size increment of ≥6 to maintain the filter structure.
- For size 27x27, this corresponds to σ=1.2x3 = 3.6



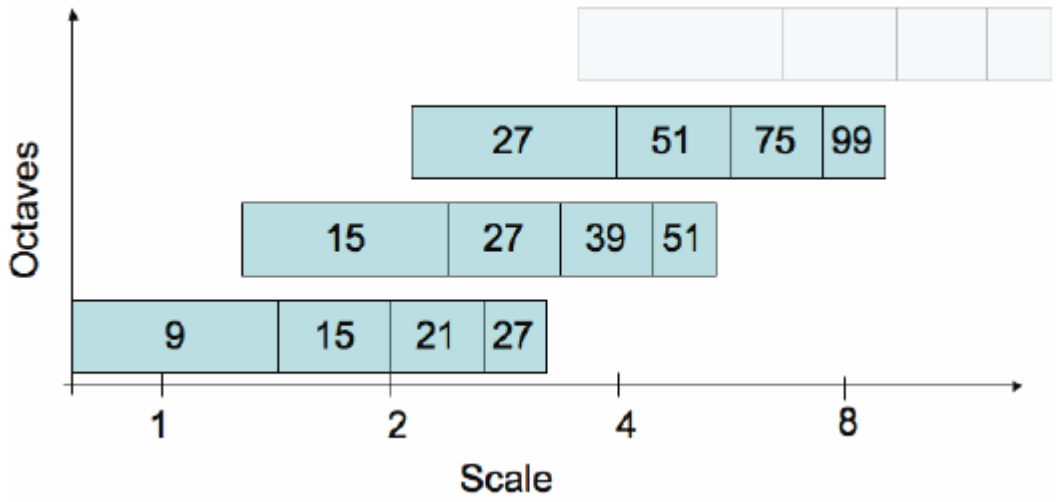
Scale Space



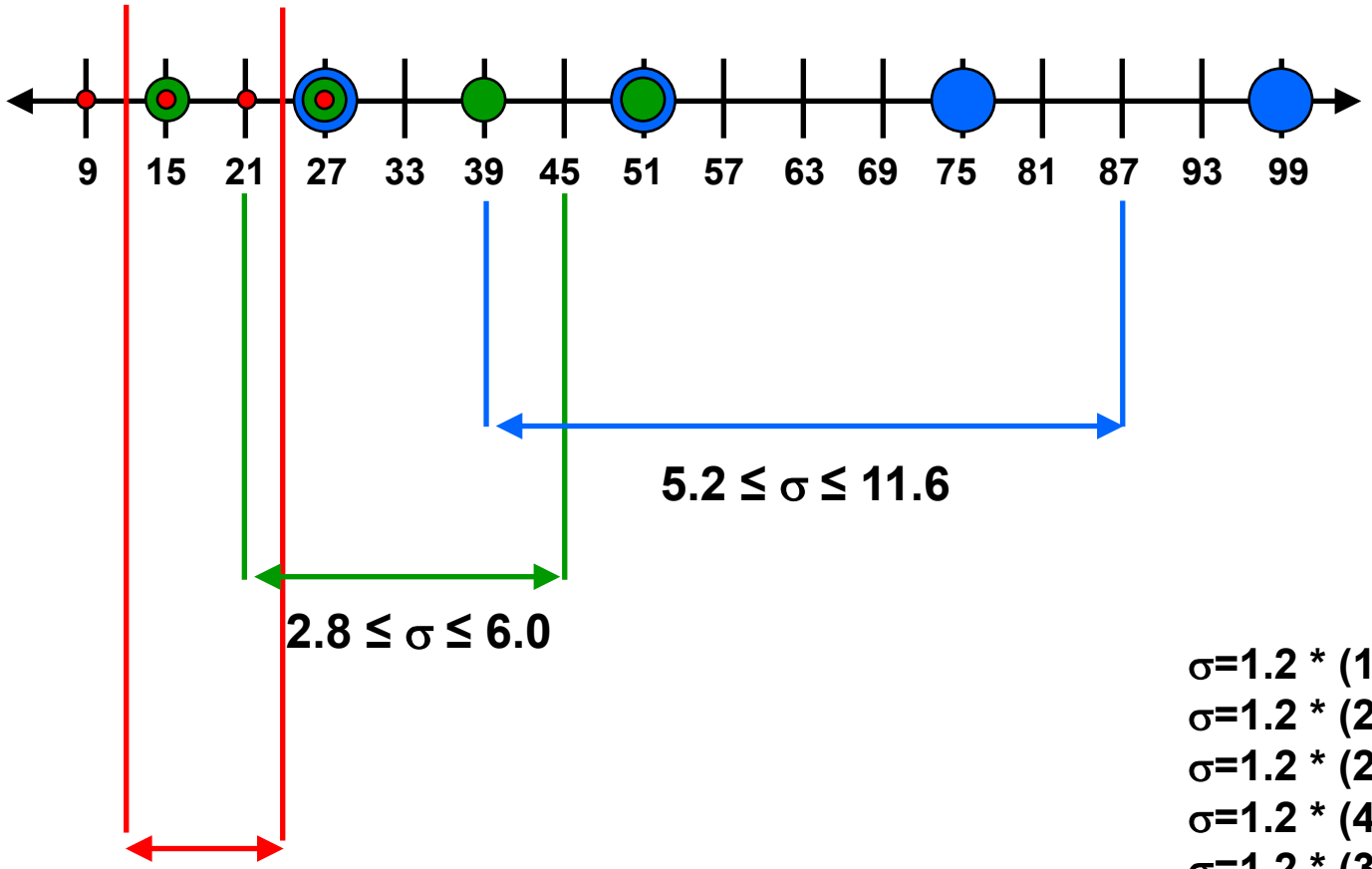
Range is exactly one octave!

Scale Space

- First octave filter sizes: 9, 15, 21, 27
- Second octave sizes: 15, 27, 39, 51
  - Increase by 12 each time (not 6)
  - Spans from 21 (σ=1.2*21/9=2.8)
    to 45 (σ=1.2*45/9=6)
    (some overlap with first octave)
- Third octave sizes: 27, 51, 75, 99
  - Increase by 24
    each time
  - Spans from
    39 (σ=1.2*39/9=5.2)
    to 87 (σ=1.2*87/9=11.6)



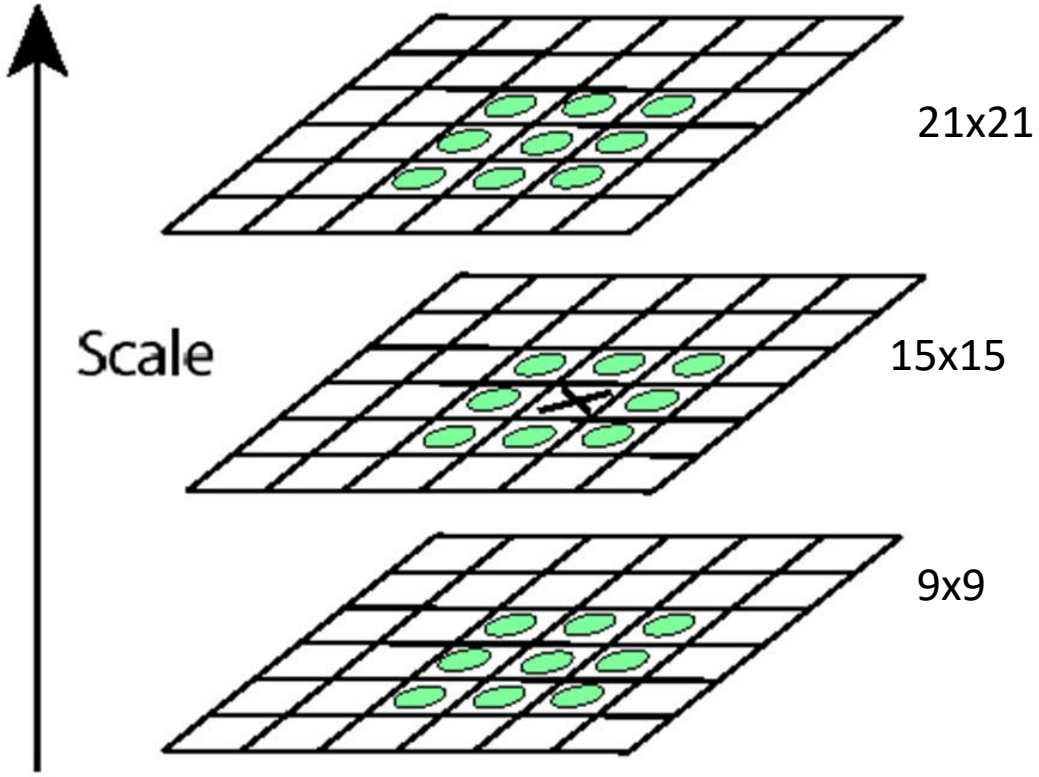
SURF Octave Overview



σ=1.2 * (12/9) = 1.6
σ=1.2 * (24/9) = 3.2
σ=1.2 * (21/9) = 2.8
σ=1.2 * (45/9) = 6.0
σ=1.2 * (39/9) = 5.2
σ=1.2 * (87/9) = 11.6

Non-Maximal Suppression

- Similar to SIFT, the point will only be selected if it is greater than 26 neighbours in x, y and σ



Feature Points Example

1st octave 9,15,21,25

2nd octave 15, 27, 39, 51.

Image credit: B. Alsadik

Feature Points Example

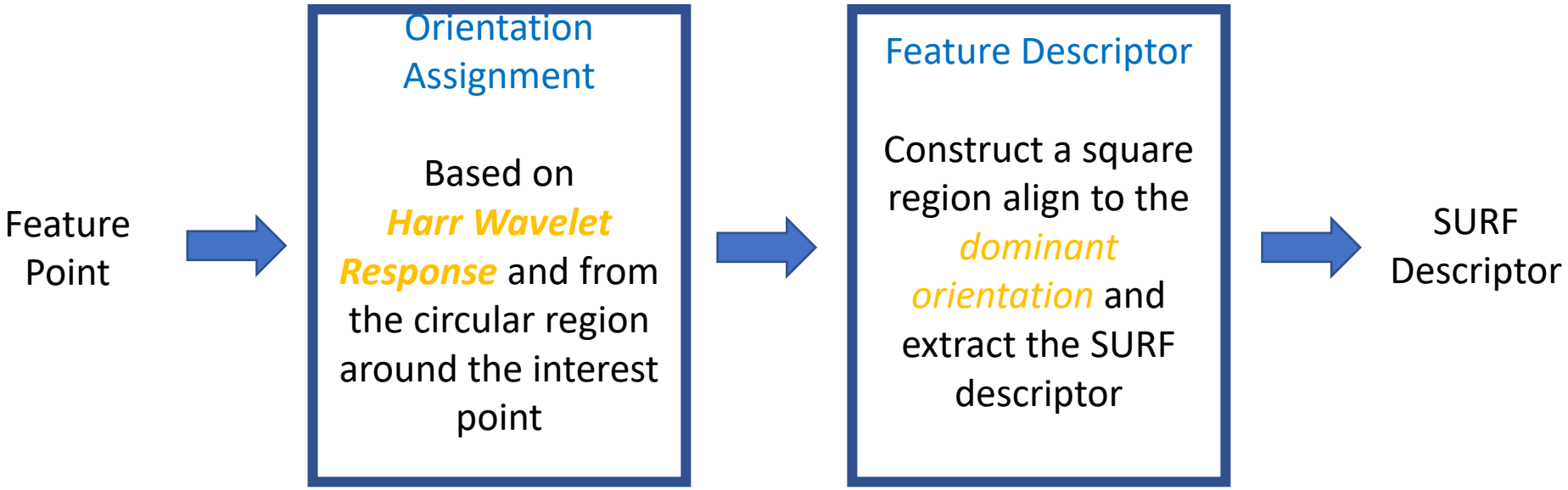
3rd octave 27, 51, 75, 99

4th octave 51, 99, 147, 195.

Image credit: B. Alsadik

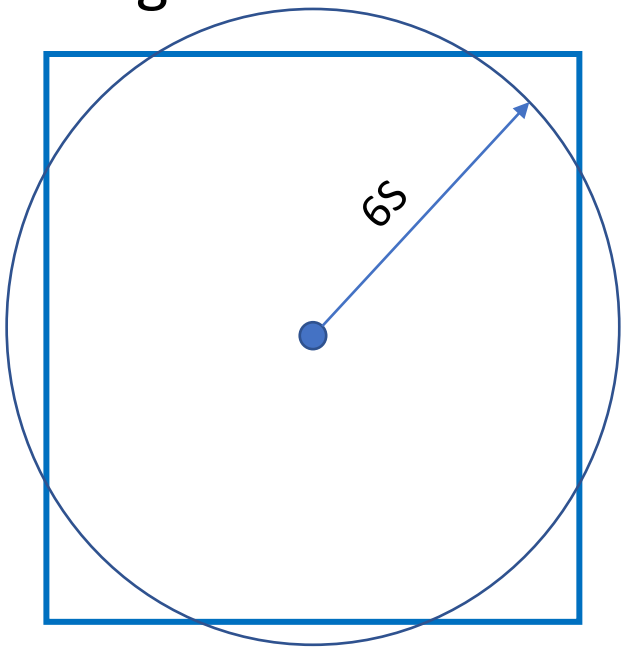
Feature Descriptor

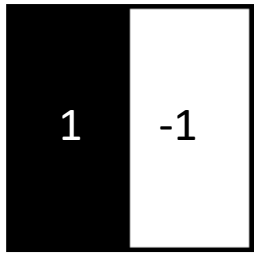
- Two Steps



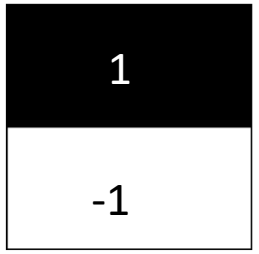
Orientation Assignment

- Construct a circle with radius r=6s around the interest point
- Apply Haar wavelet responses of size 4s to calculate the responses
- Sum all responses within a sliding orientation windows covering angle of 60 Deg.

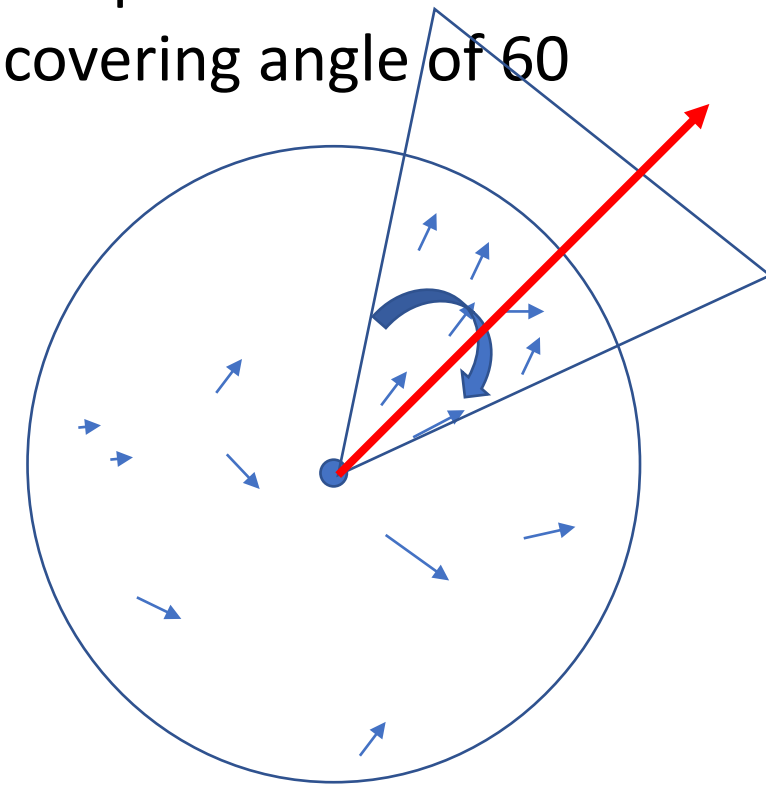




X direction response

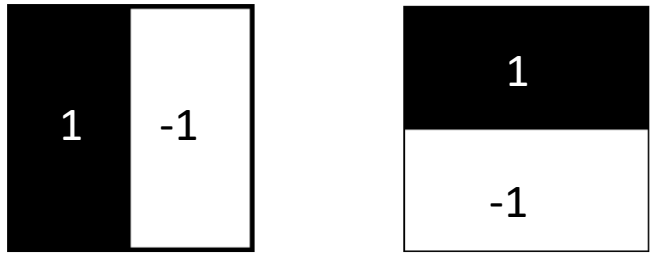


y direction response



Speed up by integral image

- We can use integral image for fast filtering. The Haar Wavelet operator only requires 6 operations to compute the responses in x or y direction at any scale.

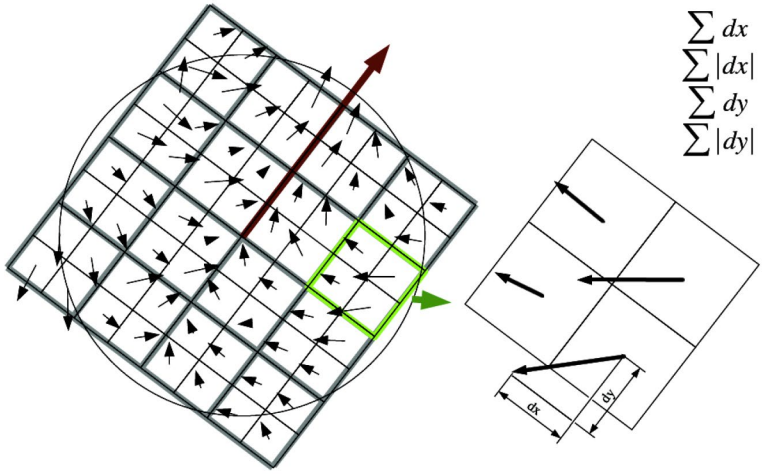


X direction response

y direction response

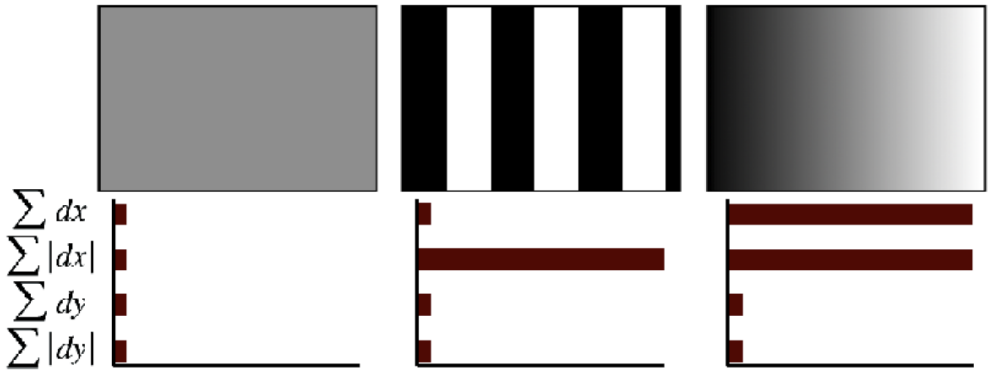
Descriptor

- Once the orientation is designed. We can now define the descriptor
  1. Split the interest region (20s x 20s) up to 4 x 4 square sub regions.
  2. Calculate the Haar wavelet responses d_x and d_y
  3. Sum the responses over each sub-region d_x and d_y as well as the absolute value of d_x and d_y
  4. Normalize the vector into unit length
  5. The resulting descriptor is of size 4 x 4 x 4 = 64



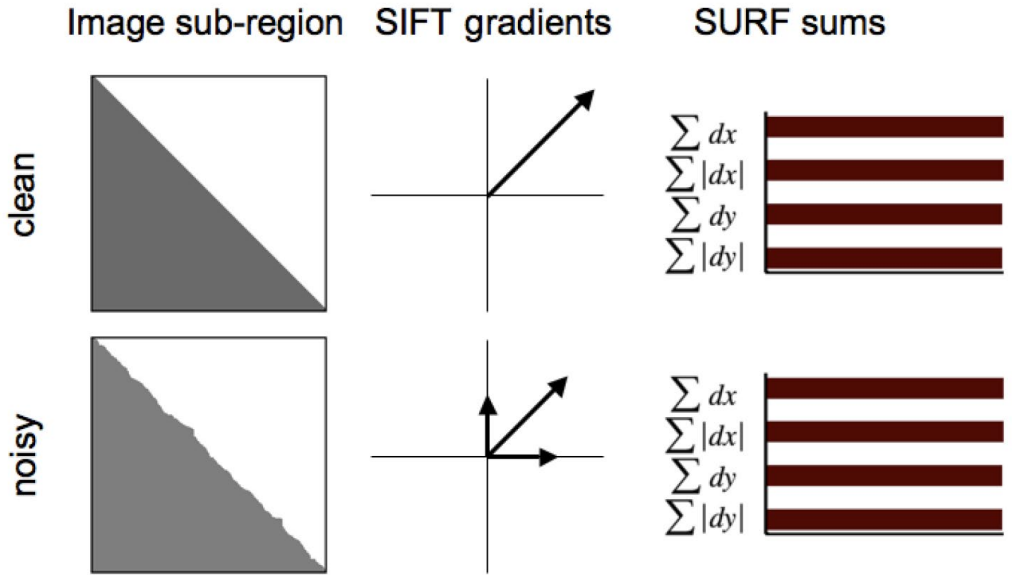
Why sum the absolute value?

- The image below shows the properties of the descriptor of three distinctively different image intensity pattern.
- Sum of absolute value results in distinctive descriptor



Another Advantages of SURF

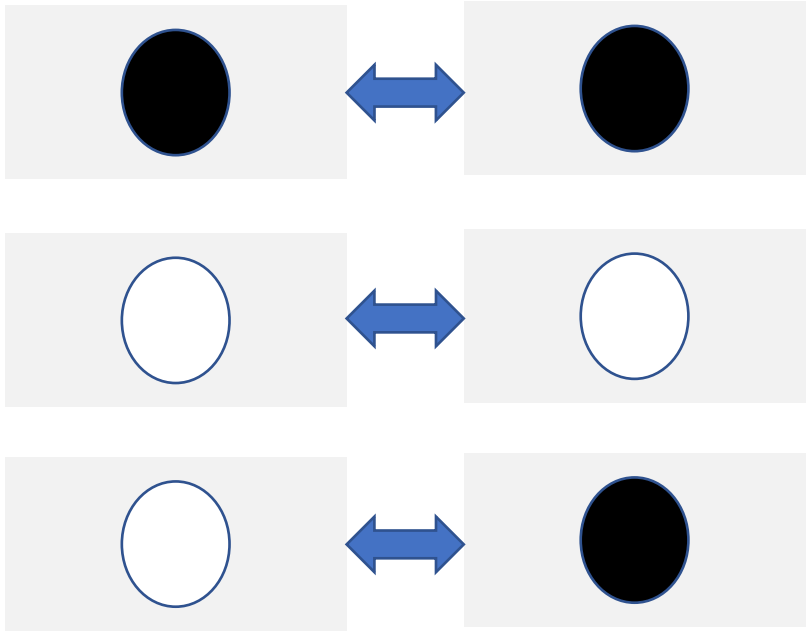
- SIFT and SURF both focus on spatial distribution of gradient information.
- SURF is more robust to noisy due to the use of integral image.



Fast Indexing for Matching

- Fast indexing through the sign of the Laplacian.
- The sign of the trace of the Hessian matrix

$$Trace = L_{xx} + L_{yy}$$



Match

Match

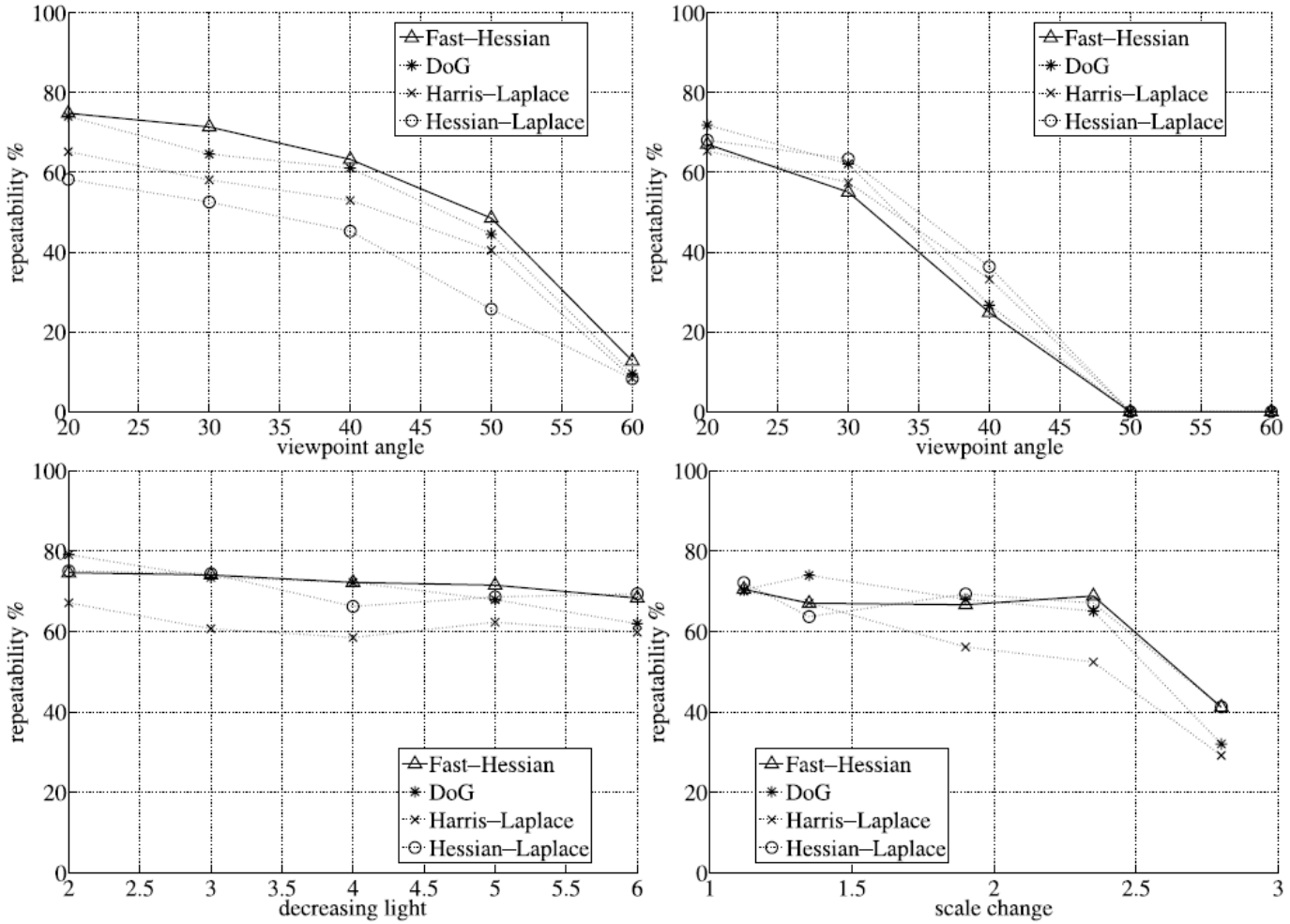
Do Not Match

Testing of SURF Detector

- The keypoint repeatability test for
  - View point change
  - Lighting change
  - Zoom and Rotation

An example image from the reference set (left) and the test set (right). Note the difference in viewpoint and colours.

Result


repeatability %
viewpoint angle
Fast–Hessian
DoG
Harris–Laplace
Hessian–Laplace
repeatability %
viewpoint angle
Fast–Hessian
DoG
Harris–Laplace
Hessian–Laplace
repeatability %
decreasing light
Fast–Hessian
DoG
Harris–Laplace
Hessian–Laplace
repeatability %
scale change
Fast–Hessian
DoG
Harris–Laplace
Hessian–Laplace

Number of key-points

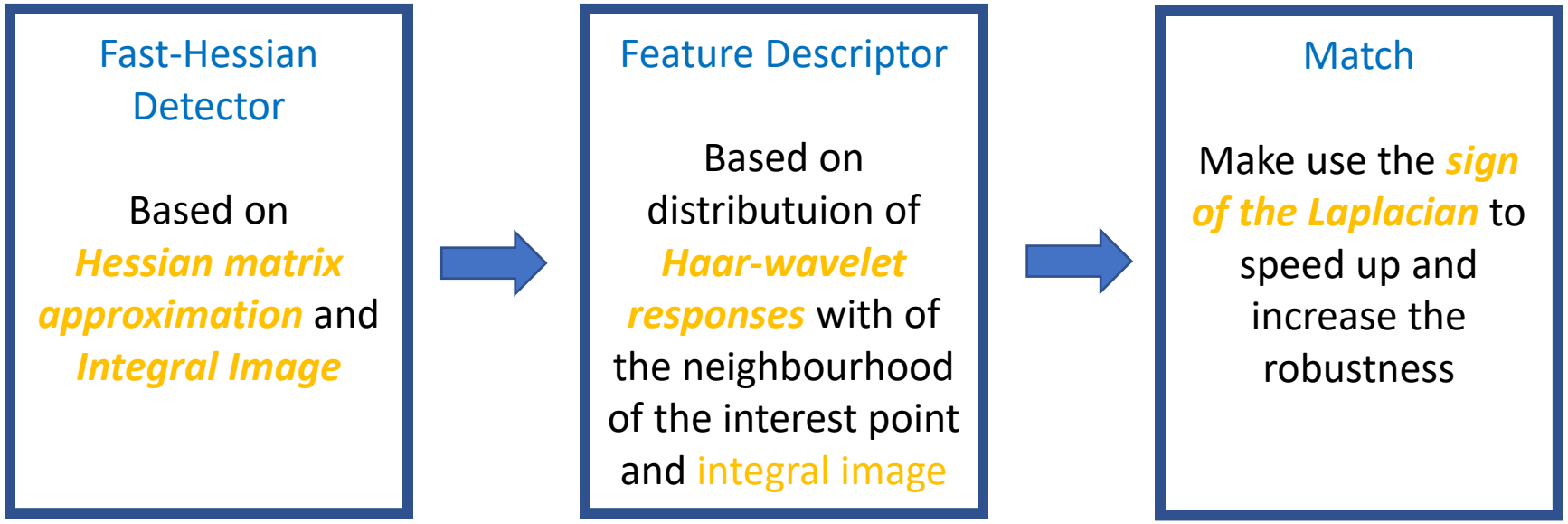
detector	threshold	nb of points	comp. time (msec)
Fast-Hessian	600	1418	120
Hessian-Laplace	1000	1979	650
Harris-Laplace	2500	1664	1800
DoG	default	1520	400

Conclusion

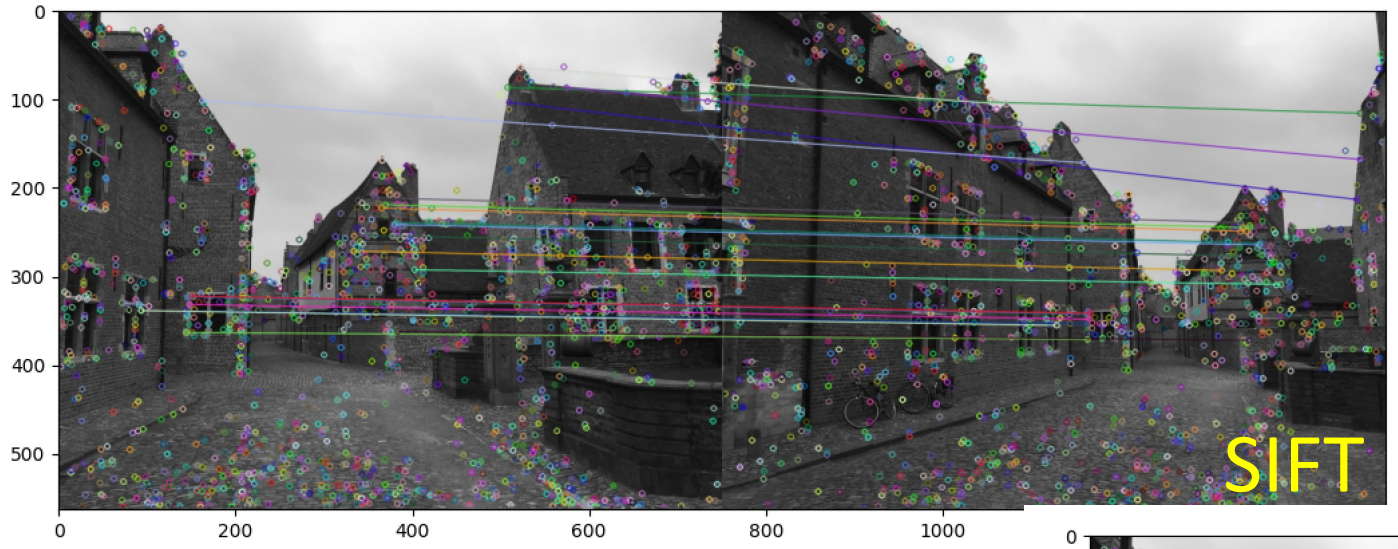
- SURF is a *fast interest point description* and *detection* scheme
- SURF is *3 times faster than SIFT* and has comparative performance
- SURF is good at handling image with *blurring* or *rotation*
- SURF is poor at handling image with *viewpoint changes*

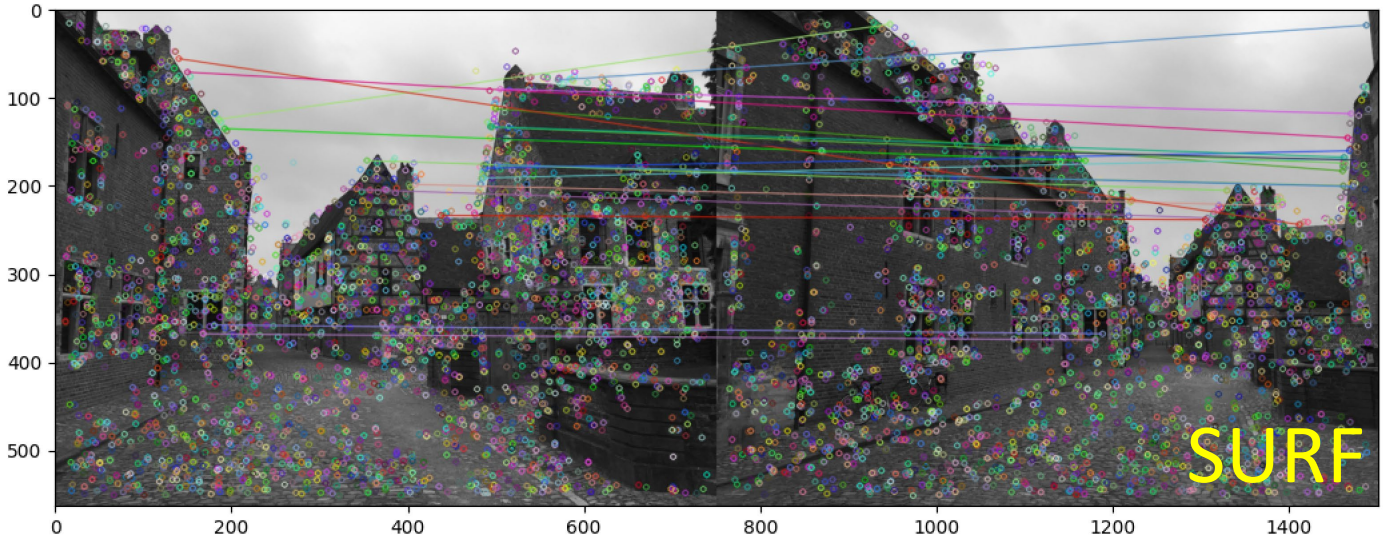
Summarize Speeded Up Robust Features (SURF)

- Three Steps in SURF



Result of SURF and SIFT



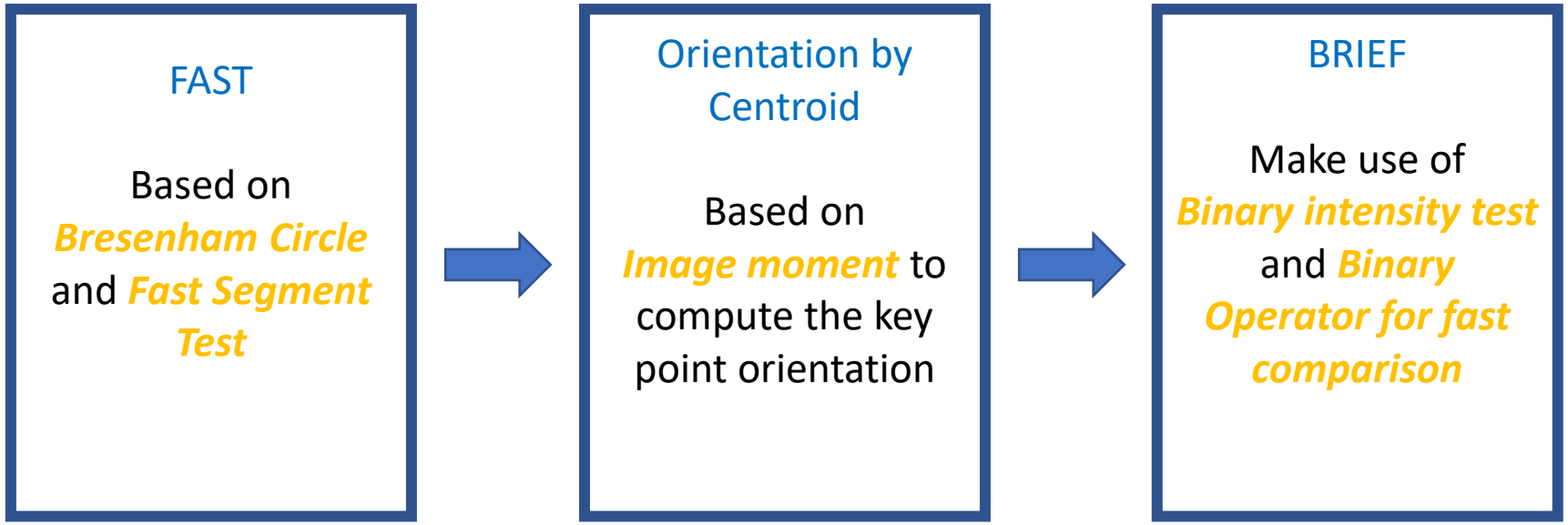


Today's Agenda

- Shape Descriptor
  - Scale Invariant Feature Transform (SIFT)
  - Speed Up Robust Feature (SURF)
  - Oriented FAST and Rotated BRIEF (ORB)
  - Histogram of Oriented Graph (HOG)

- Visual Bags of Words.

ORB Feature Descriptor



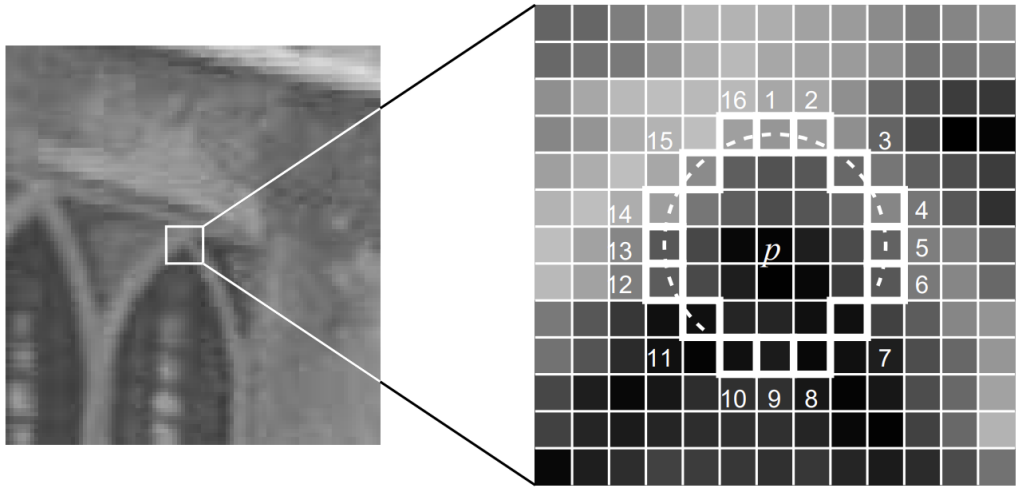
Feature from Accelerated Segments Test (FAST)

- Steps:

1. Select a pixel ***p*** in the image and the intensity of ***p*** is I_p
2. Define a Bresenham circle of radius 3 around ***p.*** A set of pixel $x \in \{1 \dots 16\}$ pixels is defined
3. Set the threshold value t***.***
4. Two States defined

$$S_{bright} = \{x | I_{p \to x} \geq I_p + t\}$$
$$S_{dark} = \{x | I_{p \to x} \leq I_p - t\}$$

1. ***p*** is a ***corner*** if there exist a set of n contiguous pixels has states of S_{bright} or S_{dark}, n=12 is chosen.



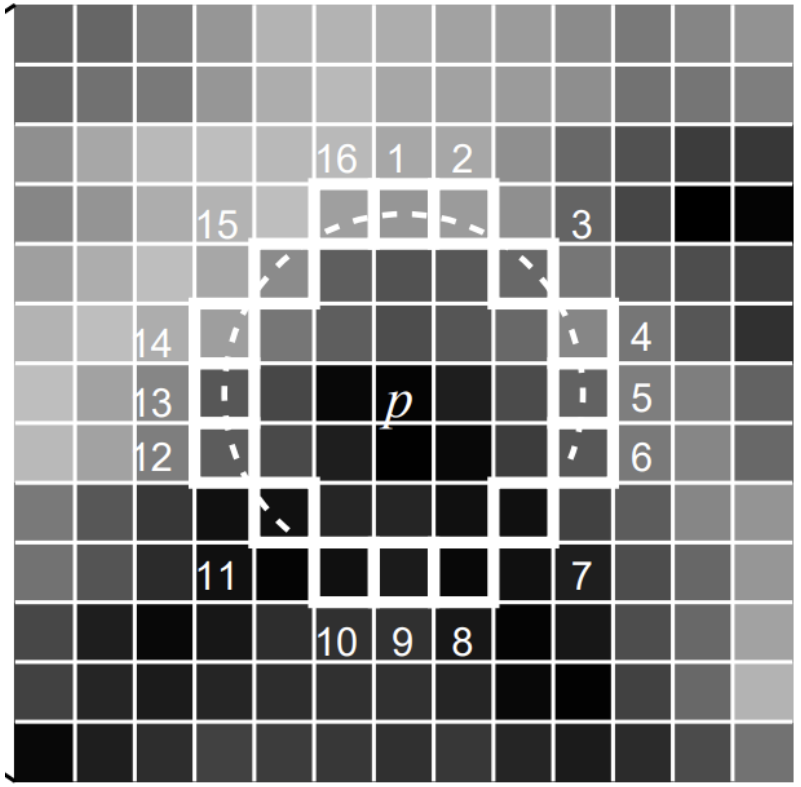
E. Rosten and T. Drummond. Machine learning for highspeed corner detection. In *European Conference on Computer Vision*, volume 1, 2006. 1

High Speed Corner Detection

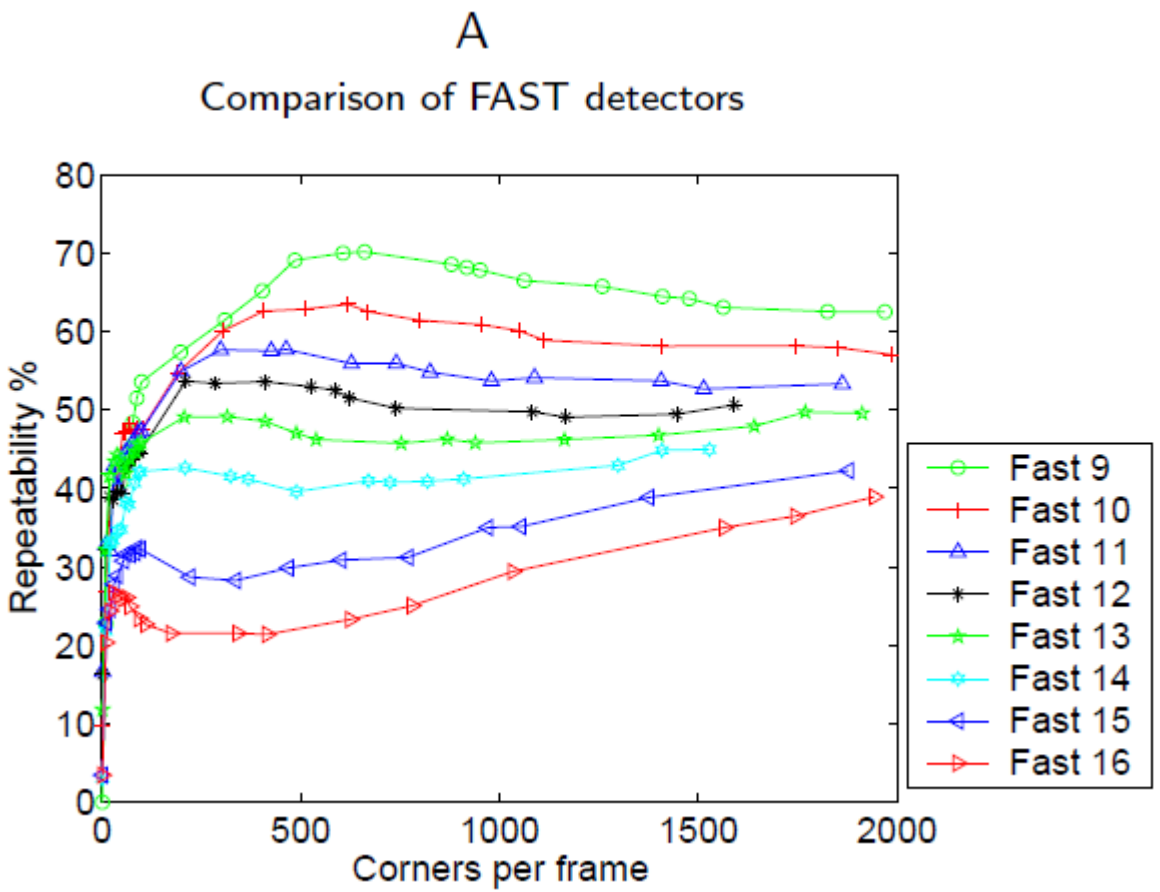
- Steps:

1. Compare intensity of pixel 1, 5, 9, 13 of the circle with I_p.
2. If ***NOT*** at least 3 pixels and darker or lighter than P, P is not an interest point (corner)
3. If at least 3 pixels and darker or lighter than P, test the rest of point.
4. Repeat 1~3 for all pixels
5. Perform non maximal suppression to select corners with good response. The response is defined by the absolute difference between the pixels in the contiguous arc and the centre pixel

$$V = \max\left(\sum_{x \in S_{bright}} \left|I_{p \to x} - I_p\right| - t\,,\ \sum_{x \in S_{dark}} \left|I_p - I_{p \to x}\right| - t\right)$$



Performance for different n



Why is it FAST?

1. FAST applies ***binary test***

$$S_{bright} = \{x | I_{p \to x} \geq I_p + t\}$$

$$S_{dark} = \{x | I_{p \to x} \leq I_p - t\}$$

2. No need to touch *all* pixels
3. Doesn't rely on integral images.
4. Good at locating possible corner candidates
5. However, it is NOT multi-scale.

Speed Comparison

Detector	Opteron 2.6GHz		Pentium III 850MHz	
	ms	%	ms	%
Fast $n = 9$ (non-max suppression)	1.33	6.65	5.29	26.5
Fast $n = 9$ (raw)	1.08	5.40	4.34	21.7
Fast $n = 12$ (non-max suppression)	1.34	6.70	4.60	23.0
Fast $n = 12$ (raw)	1.17	5.85	4.31	21.5
Original FAST $n = 12$ (non-max suppression)	1.59	7.95	9.60	48.0
Original FAST $n = 12$ (raw)	1.49	7.45	9.25	48.5
Harris	24.0	120	166	830
DoG	60.1	301	345	1280
SUSAN	7.58	37.9	27.5	137.5

Table 1. Timing results for a selection of feature detectors run on fields (768×288) of a PAL video sequence in milliseconds, and as a percentage of the processing budget per frame. Note that since PAL and NTSC, DV and 30Hz VGA (common for web-cams) have approximately the same pixel rate, the percentages are widely applicable. Approximately 500 features per field are detected.

Orientation of Intensity Centroid

- Similar to the concept of moment. The moment is defined by

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

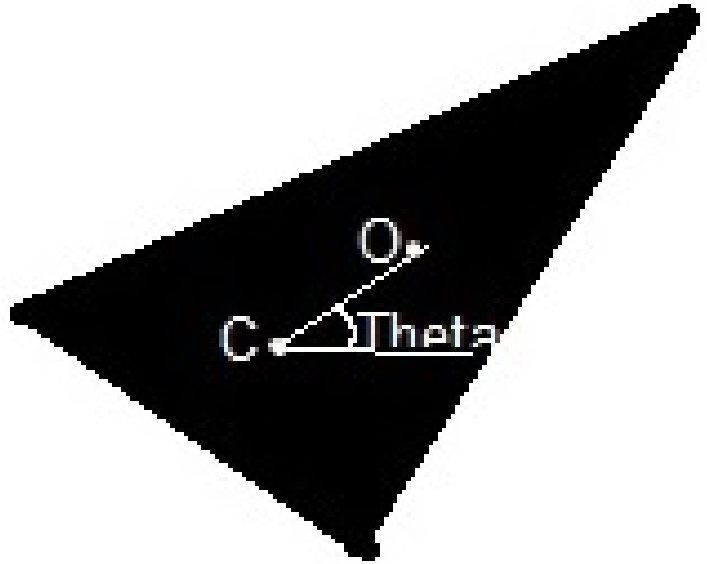
- The centroid is defined by

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{oo}}\right)$$

Construct a vector OC and the orientation of the patch is defined as

$$\theta = arctan2(m_{01}, m_{10})$$ atan2 – Wikipedia

The patch is rotated by θ making it rotation invariant



P. L. Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291 – 307, 1999

Binary Robust Independent Elementary Features(BRIEF)

- Define a test τ on patch **p**

$$\tau(p; x, y) := \begin{cases} 1 & if\ p(x) < p(y) \\ 0 & otherwise \end{cases}$$

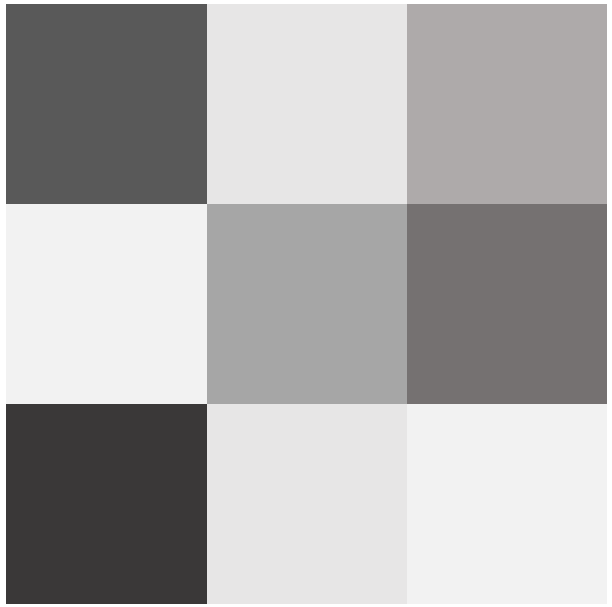
- Where p(x) is the pixel intensity .

- Choosing a set of $n_d(x, y)$ location pairs uniquely on the patch. Concatenate the test result to a n_d dimensional bitstring

$$f_{nd}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1}\, \tau(p; x, y)$$

Example

Image



1	2	3
4	5	6
7	8	9

Index

- Pairs: {(1,6), (2,8), (3,5), (1,9), (8,3)}
- Test: τ=1 τ=0 τ=0 τ=1 τ=0
- $f = \{10010\}$

$$\tau(p; x, y) \coloneqq \begin{cases} 1 & if\ p(x) < p(y) \\ 0 & otherwise \end{cases}$$

Key Advantages of Binary Descriptor

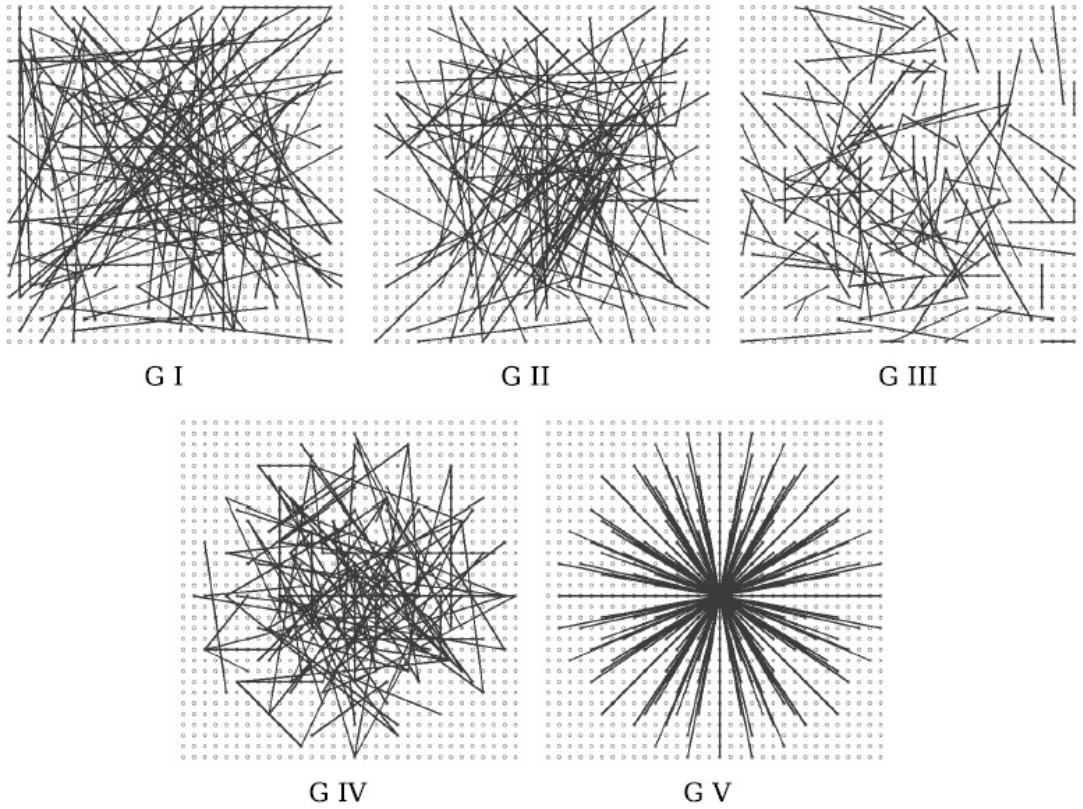
- Compact Descriptor
  - The number of pairs defines the bitstring length.
- Fast to compute
  - Simply intensity value comparison
- Fast to Compare
  - Hamming distance

$$d_{Hamming}(f_1, f_2) = SUM(XOR(f_1, f_2))$$

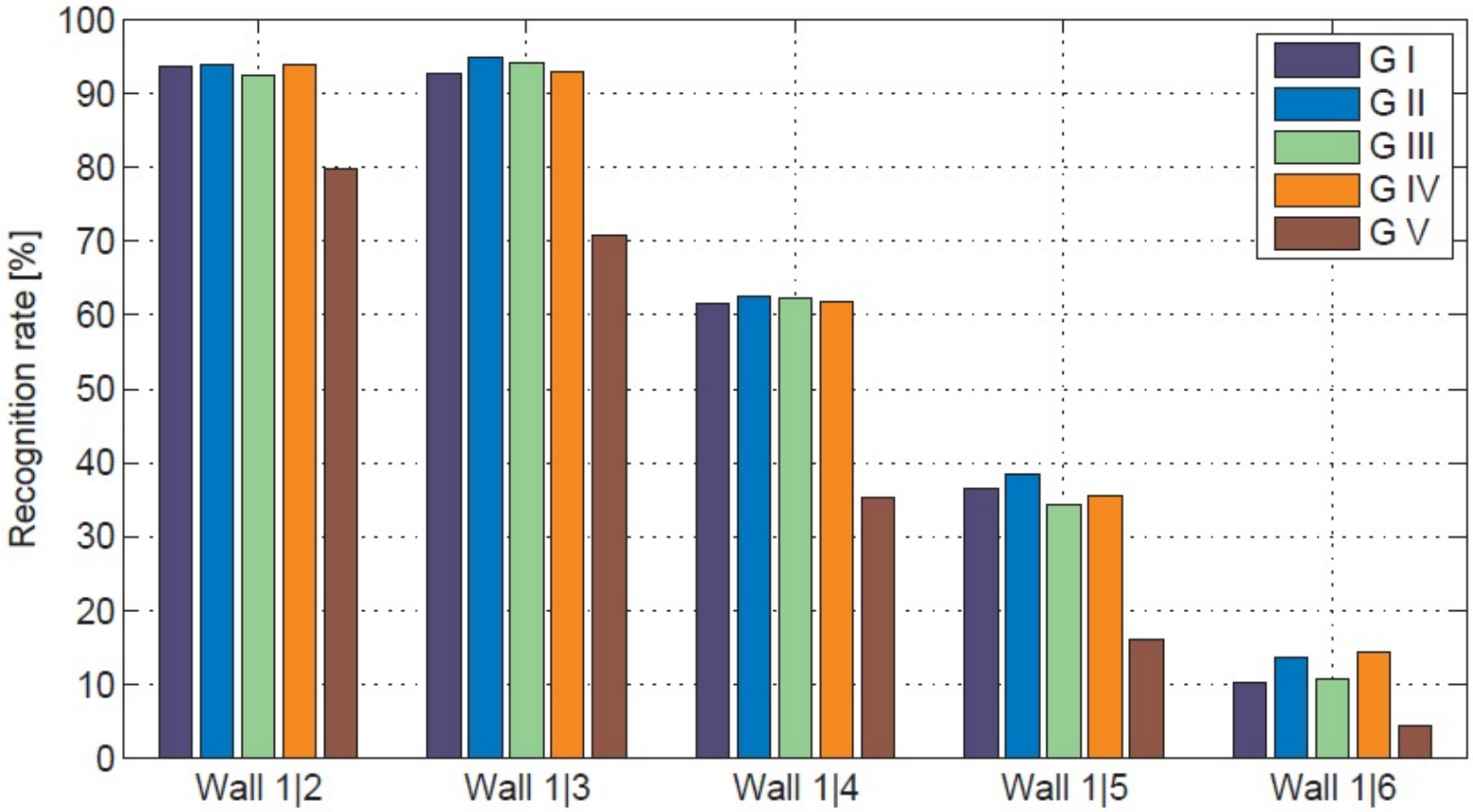
How to chose the sample point pairs?

Different Sampling Strategies

- Five Sampling Strategies
  1. . X and Y are randomly uniformly sampled
  2. X and Y are randomly sampled using a Gaussian distribution
  3. X and Y are randomly sampled with length following Gaussian distribution
  4. X and Y are randomly sampled from discrete location of a coarse polar gird.
  5. For each i, Xi is (0, 0) and Yi takes all possible values on a coarse polar grid.



Effect of different sampling technique



Performance

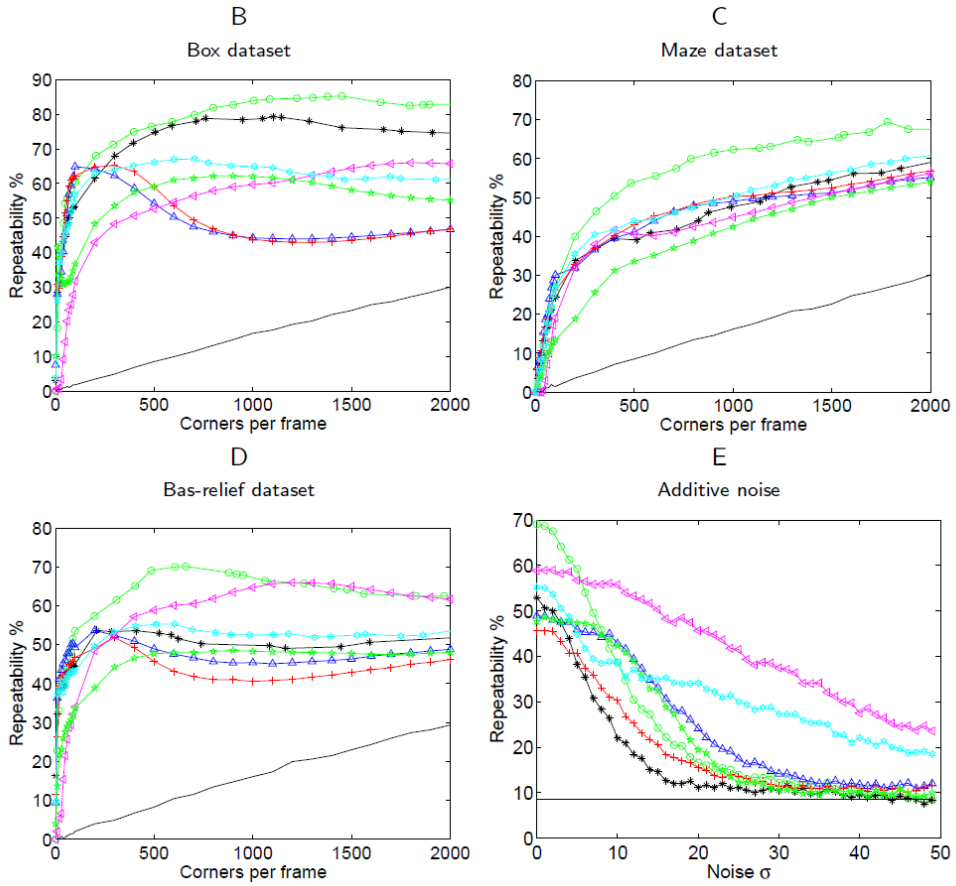


Figure 6. A: A comparison of the FAST detectors shown that $n = 9$ is the most repeatable. For $n \leq 8$, the detector starts to respond strongly to edges. B, C, D: Repeatability results for the three datasets as the number of features per frame is varied. D: repeatability results for the bas-relief data set (500 features per frame) as the amount of Gaussian noise added to the images is varied. For FAST and SUSAN, the number of features can not be chosen arbitrarily; the closest approximation to 500 features per frame achievable is used.

Summary of ORB

- The time comparison of ORB, SURF and SIFT

Detector	ORB	SURF	SIFT
Time per frame (ms)	**15.3**	217.3	5228.7

- These times were averaged over 24 640x480 images from Pascal dataset. ORB is an order of magnitude faster than SURF and two orders faster than SIFT.

Today's Agenda

- Shape Descriptor
  - Scale Invariant Feature Transform (SIFT)
  - Speed Up Robust Feature (SURF)
  - Oriented FAST and Rotated BRIEF (ORB)
  - Histogram of Oriented Graph (HOG)

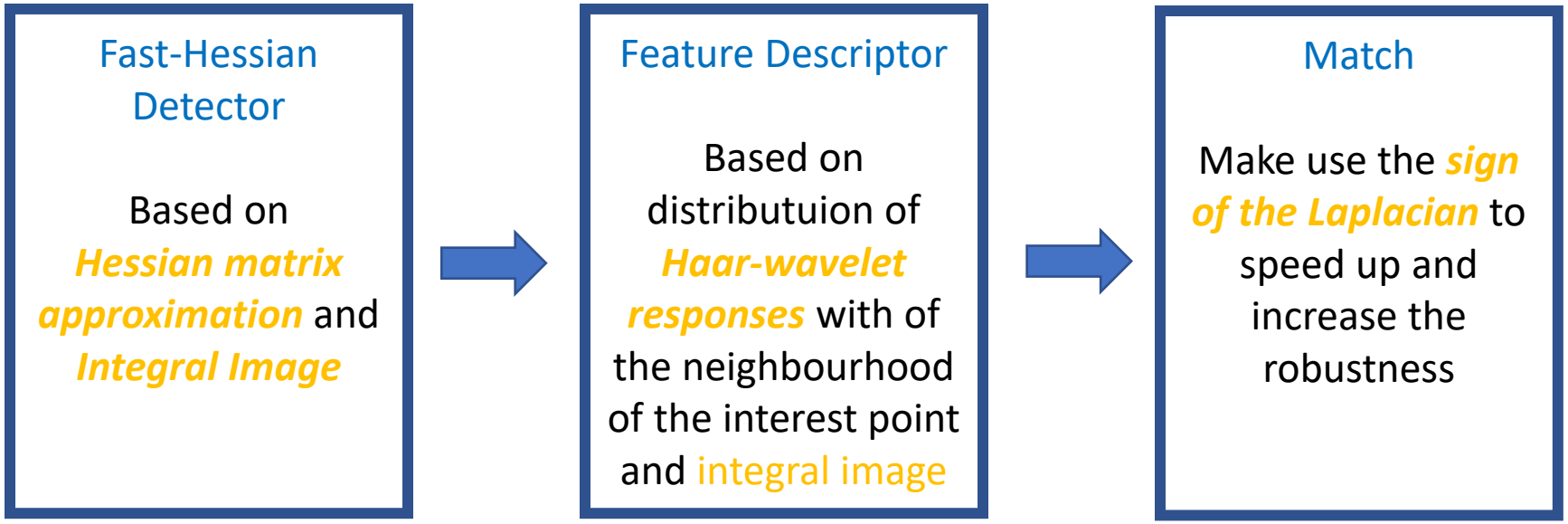
- Visual Bags of Words.

Summarizes the steps for building SIFT Descriptor

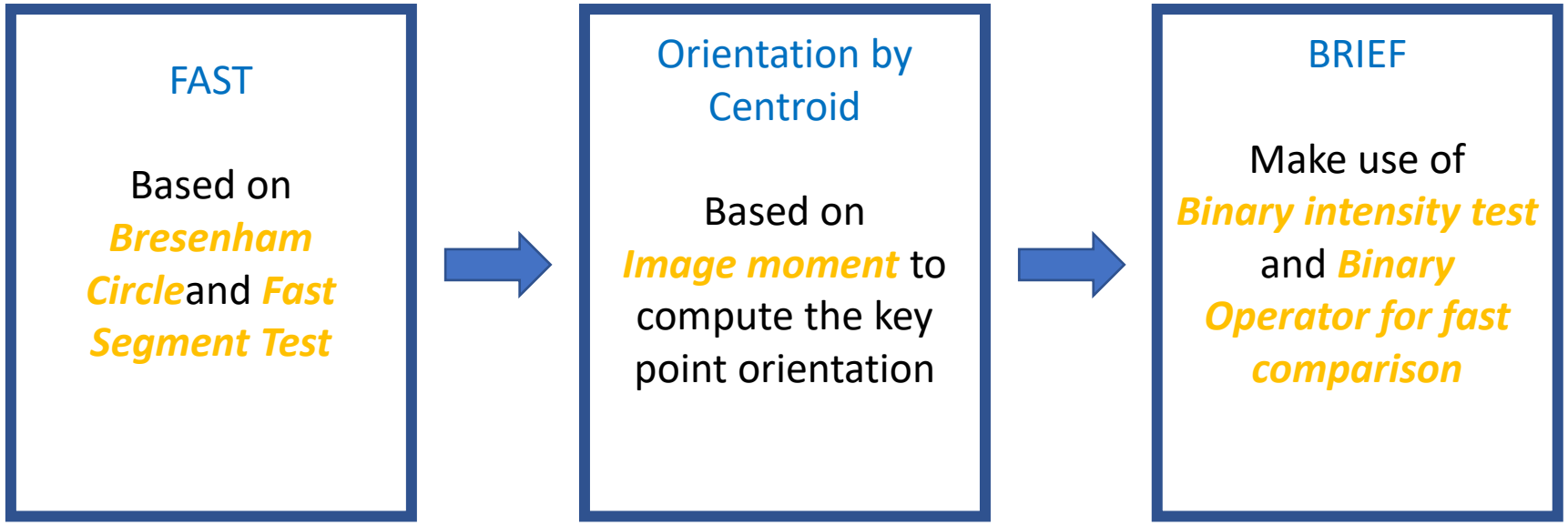
1. *Constructing a scale space* - Building the smoothing versions of the image with different σ value and image sizes (octaves).
2. *LoG Approximation by DoG* – Approximate LoG by DoG which is less computational expensive.
3. *Finding key points* - Search for maxima/minima in DoG images across scales
4. *No maxima suppression* – Eliminate the low contrast key points and edges
5. *Assigning an orientation* – The orientation of each pixel is calculated, and canonical orientation is defined at the peak. All pixels' orientation are defined based on the canonical orientation. The orientation is quantiles into 8 orientations(bins) and histogram is constructed.
6. *Generate SIFT feature description* – generate the 8-bin histogram of 4x4 regions making *128 dimensions descriptor*.

Summarizes of Speeded Up Robust Features (SURF)

- Three Steps in SURF



Summarizes: ORB Feature Descriptor



Today's Agenda

- Shape Descriptor
  - Scale Invariant Feature Transform (SIFT)
  - Speed Up Robust Feature (SURF)
  - Oriented FAST and Rotated BRIEF (ORB)
  - Histogram of Oriented Graph (HOG)

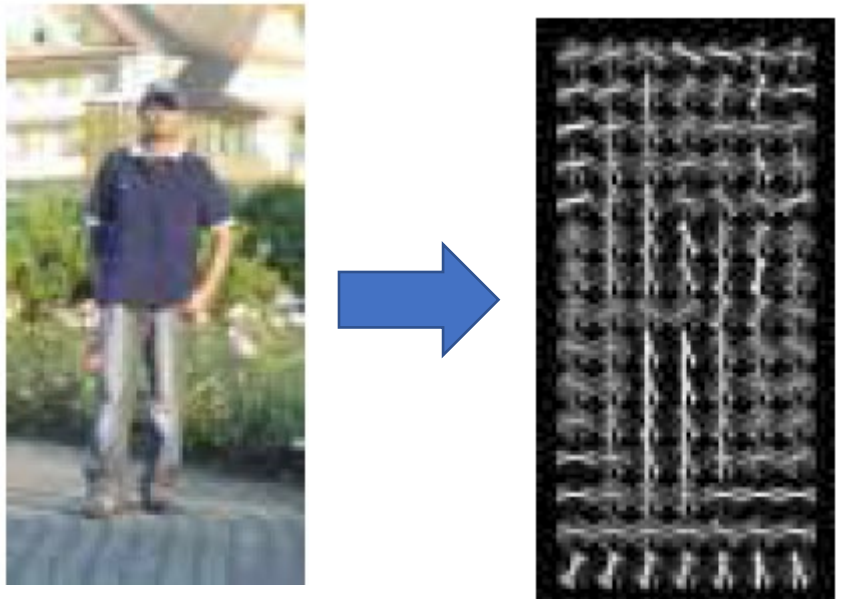
- Visual Bags of Words.

Histogram of Oriented Gradient (HOG)

Motivation: To develop a robust feature set that allows the "object" form to be discriminated cleanly, even in clustered backgrounds under difficult illumination.

HOG was firstly proposed by Dalal & Triggs in 2005 for Pedestrian detection.

HOG+SVM has been used very successfully for many object categories.

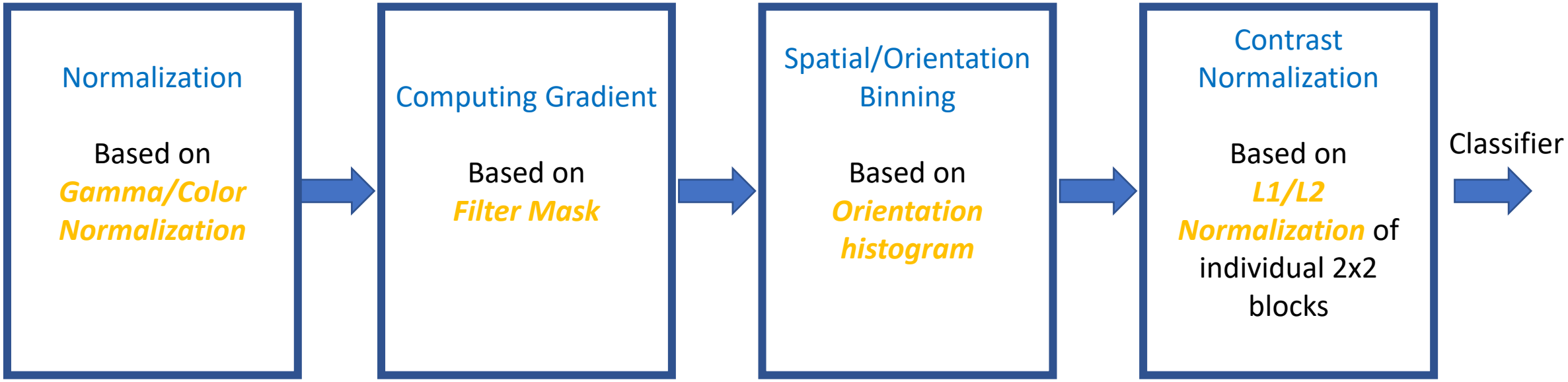


Original Image HOG Descriptor

Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision & Pattern Recognition (CVPR '05), Jun 2005, San Diego, United States. pp.886–893

Steps for HOG

- HOG has four major steps



Gamma Normalization

- Also known as Gama Correction. It is used to correct the different between camera capture content and how our eye respond to light.
- It follows a power-law

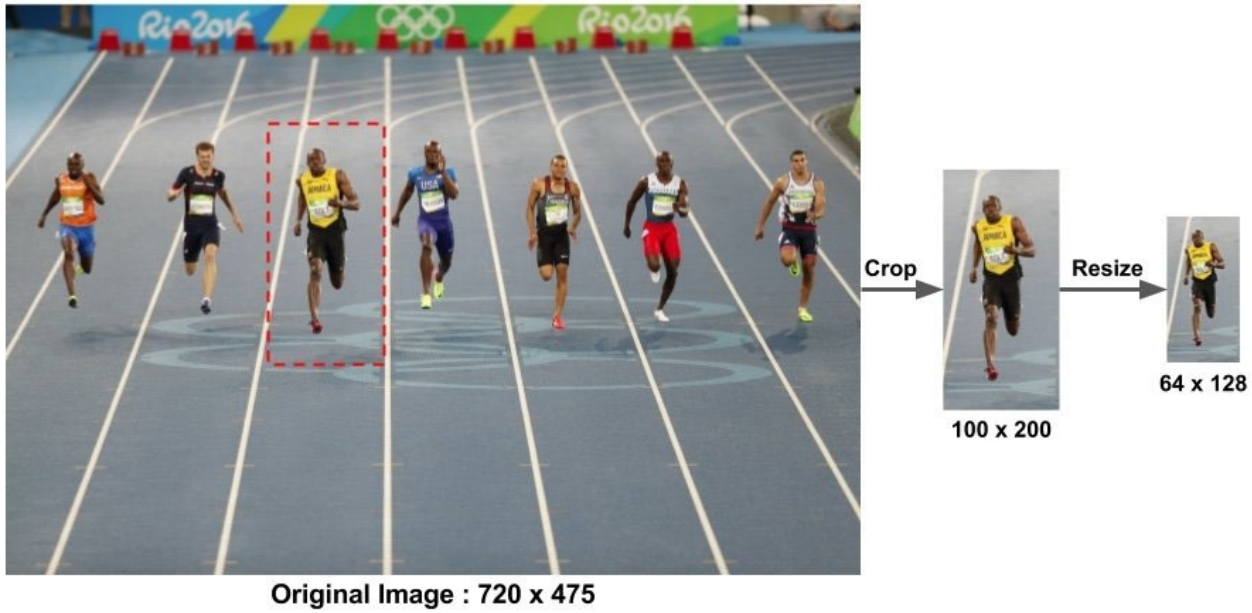
$$V_{out} = AV_{in}^{\gamma}$$

- According to the author, the normalization has only modest effect on performance due to the later descriptor normalization process.

Source: Gamma Correction - an overview | ScienceDirect Topics

Computing Gradient

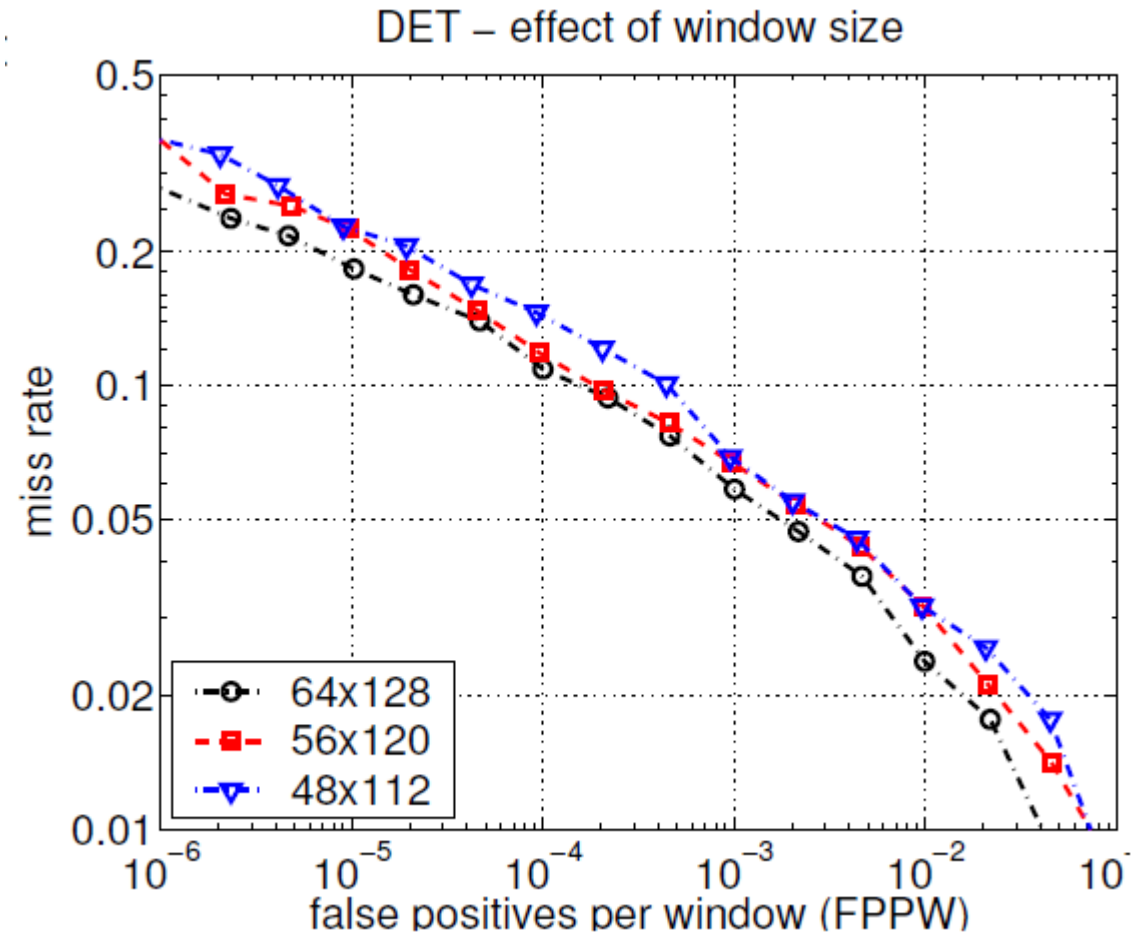
Select a patch from the image with aspect ratio of 1:2



The cropped image is resized to 64x128 pixels

Image Source: http://www.learnopenv.com

Why 64x128?


DET – effect of window size
0.5
0.2
0.1
0.05
0.02
0.01
miss rate
64x128
56x120
48x112
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
false positives per window (FPPW)

Computing Gradient

- The Gradient is calculated by horizontal and vertical derivative mask without smoothing

-1	0	1

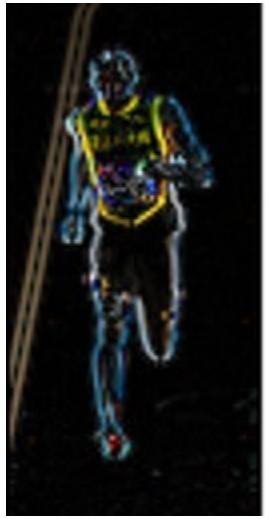
g_x

-1
0
1

g_y

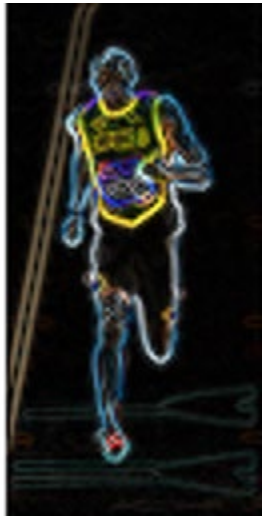
$$magnitude\ g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan\frac{g_y}{g_x}$$



x-gradient

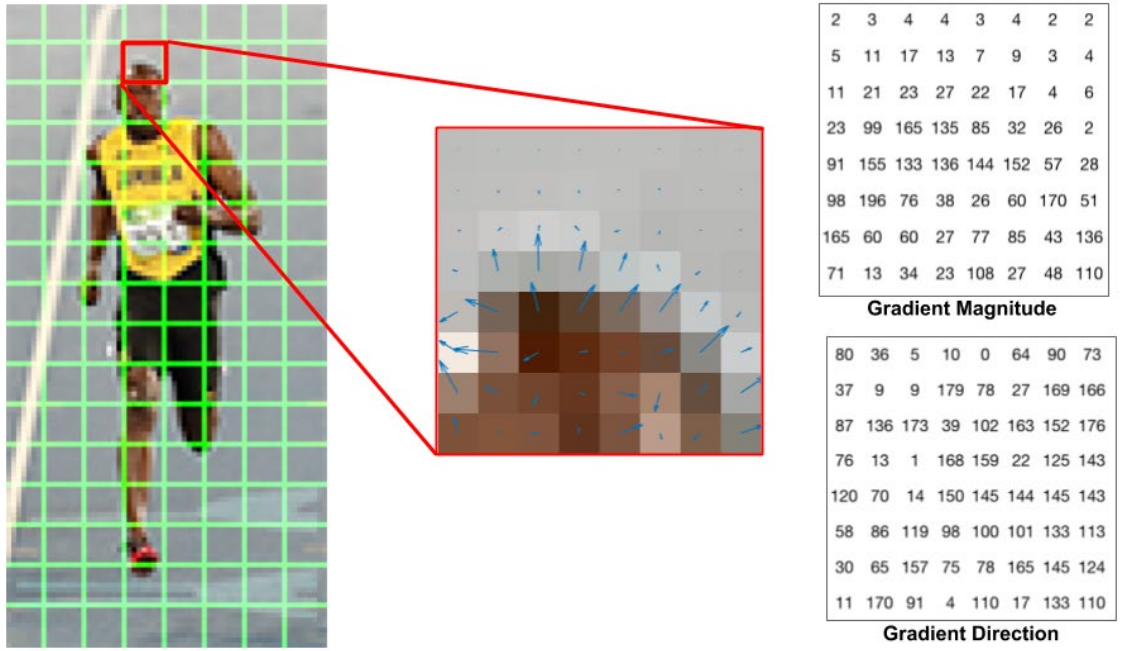
y-gradient



Magnitude of gradient

Spatial and Orientation Binning

- Divide the image into 8x8 cell
- For each cell, we have the magnitude and direction value (θ) computed
- $if\ \theta > 180^o\ then\ \theta = 180^o - \theta$
- The total size of each cell is
  - 8x8x2 (magnitude and direction)



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

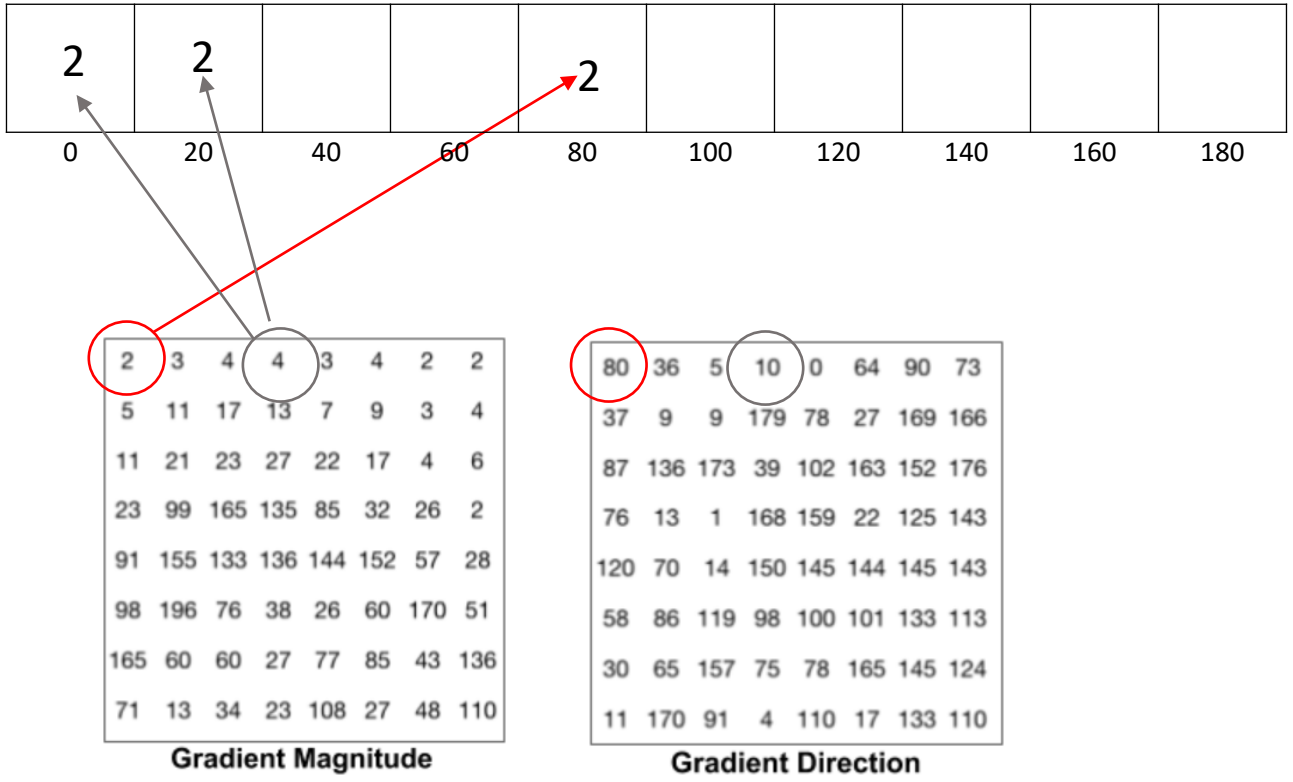
80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

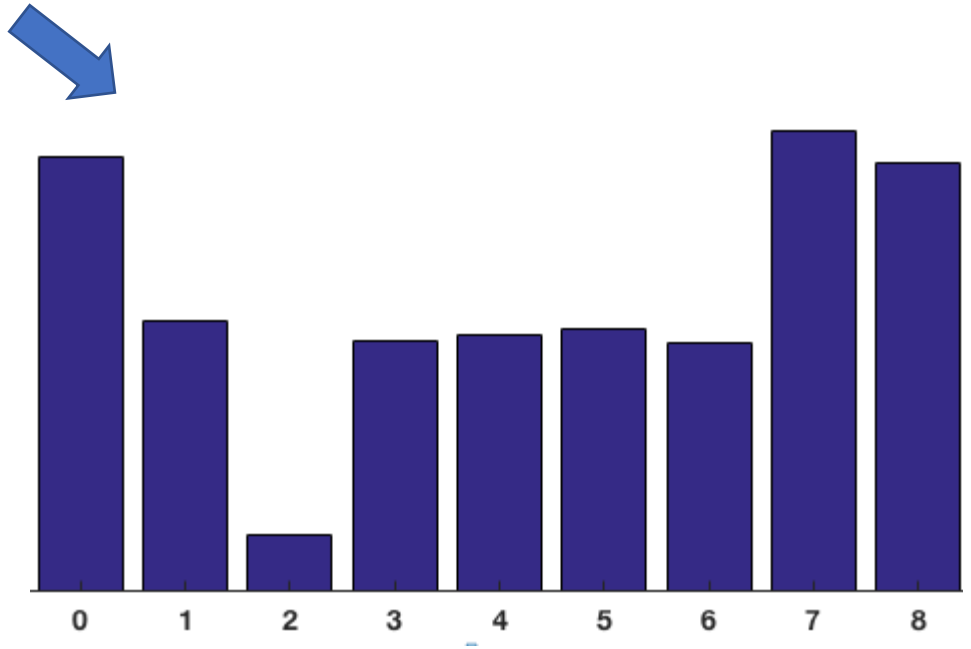
Gradient Direction

$if\ \theta > 180^o\ then\ \theta = 180^o - \theta$

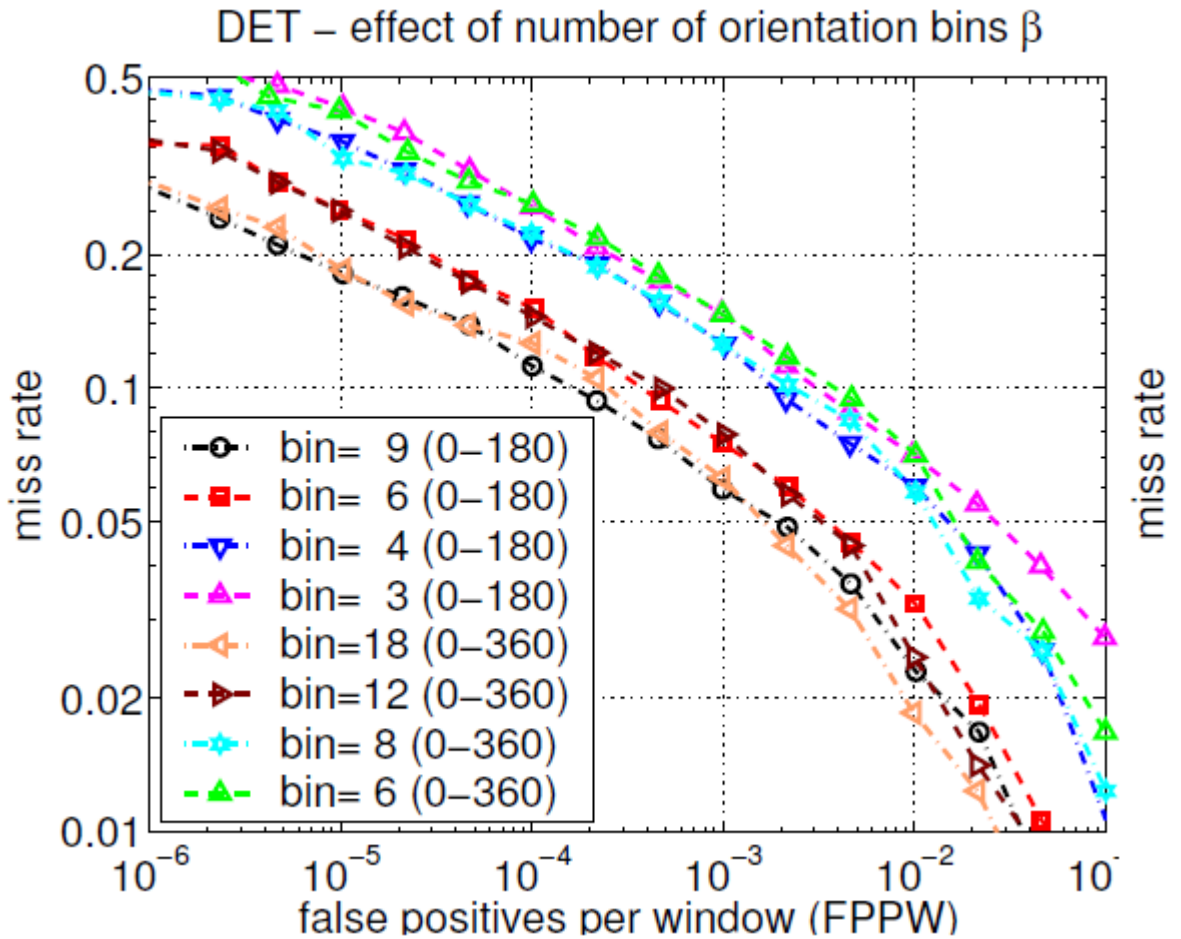
Spatial and Orientation Binning

- For each block, construct a 9 bin histogram





Why 9 bin?

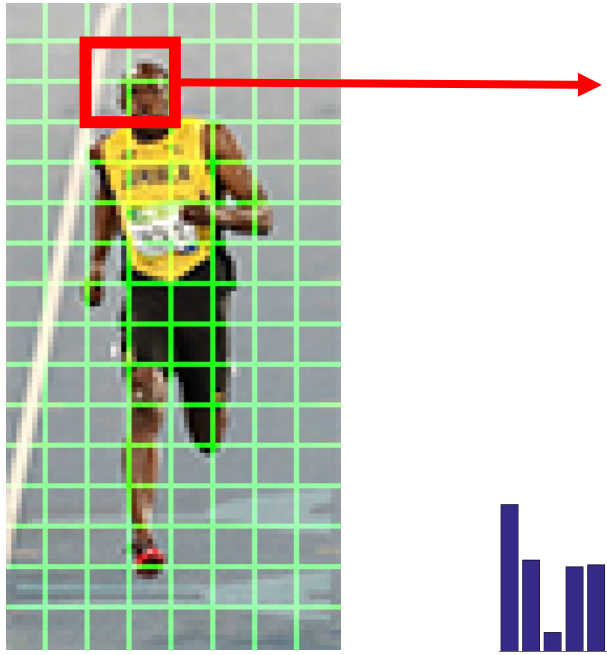

DET – effect of number of orientation bins β
miss rate
miss rate
0.5
0.2
0.1
0.05
0.02
0.01
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
false positives per window (FPPW)
bin= 9 (0–180)
bin= 6 (0–180)
bin= 4 (0–180)
bin= 3 (0–180)
bin=18 (0–360)
bin=12 (0–360)
bin= 8 (0–360)
bin= 6 (0–360)

Block Normalization

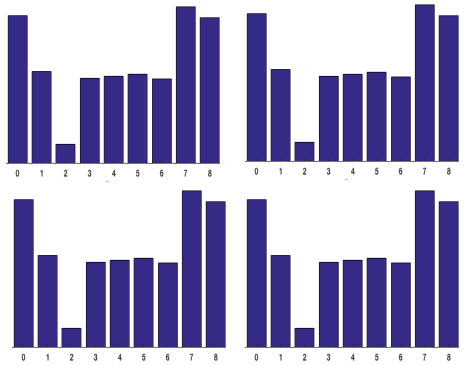
- Why Normalization?
  - Gradient strength vary over a wide range owing to local variation in illumination and foreground-background contrast.
  - We need to normalize this for good result

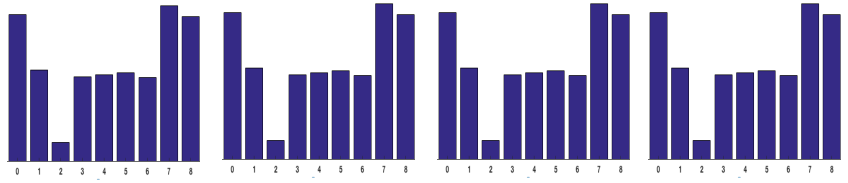
- Steps:

1. Group the cells into 4x4 cells
2. Concatenate 4 histogram into a vector $\mathbf{v}$ which is 36x1 vector



Each block yields 4x4x9 bin histogram





Concatenate to 36x1 vector

Block Normalization

- A couple of methods can be used.

L2-norm:

$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2+\epsilon^2}}$$

L2-hys:

$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_2^2+\epsilon^2}}$$ limited to v=0.2

L1-norm:

$$\mathbf{v} \rightarrow \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|_1+\epsilon}}$$

L1-sqrt:

$$\mathbf{v} \rightarrow \sqrt{\frac{\mathbf{v}}{\|\mathbf{v}\|_1+\epsilon}}$$

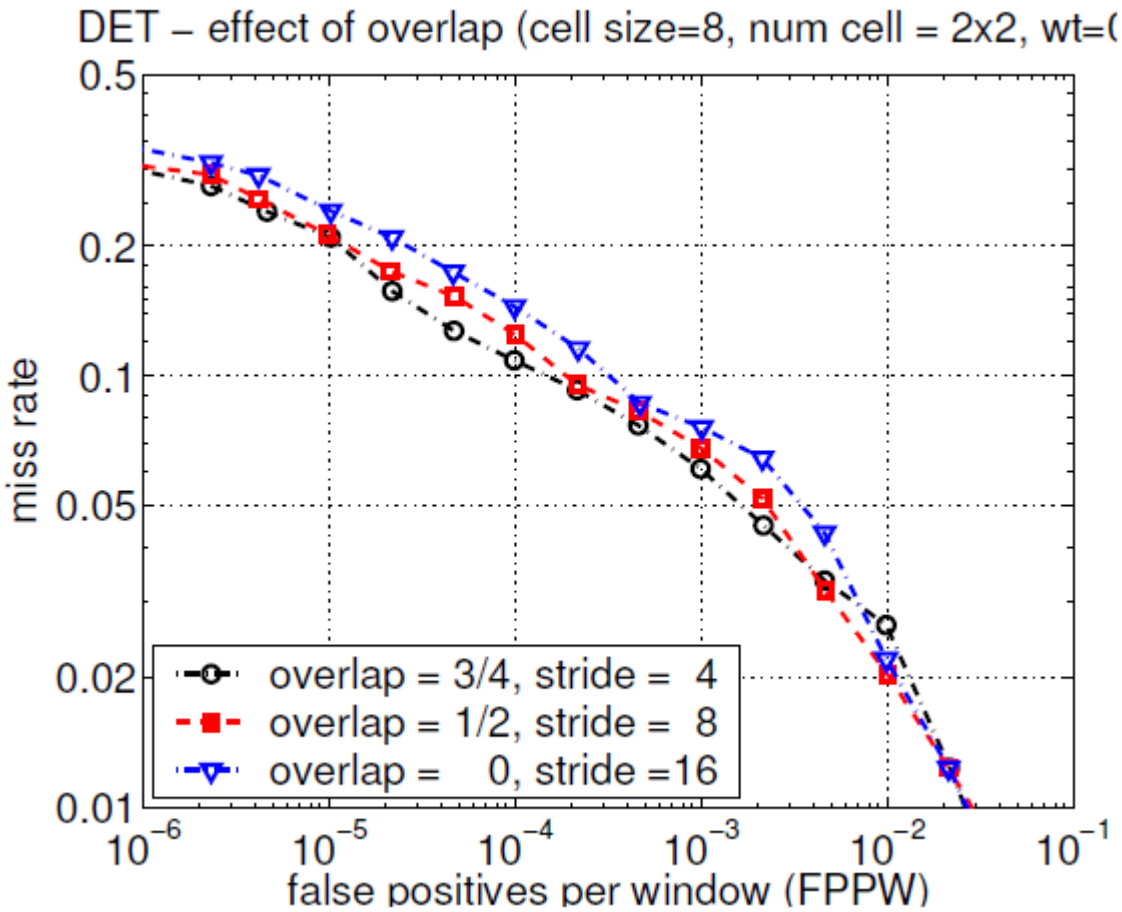
The blocks are normalized by overlapping the each others

Strike = 8 pixels

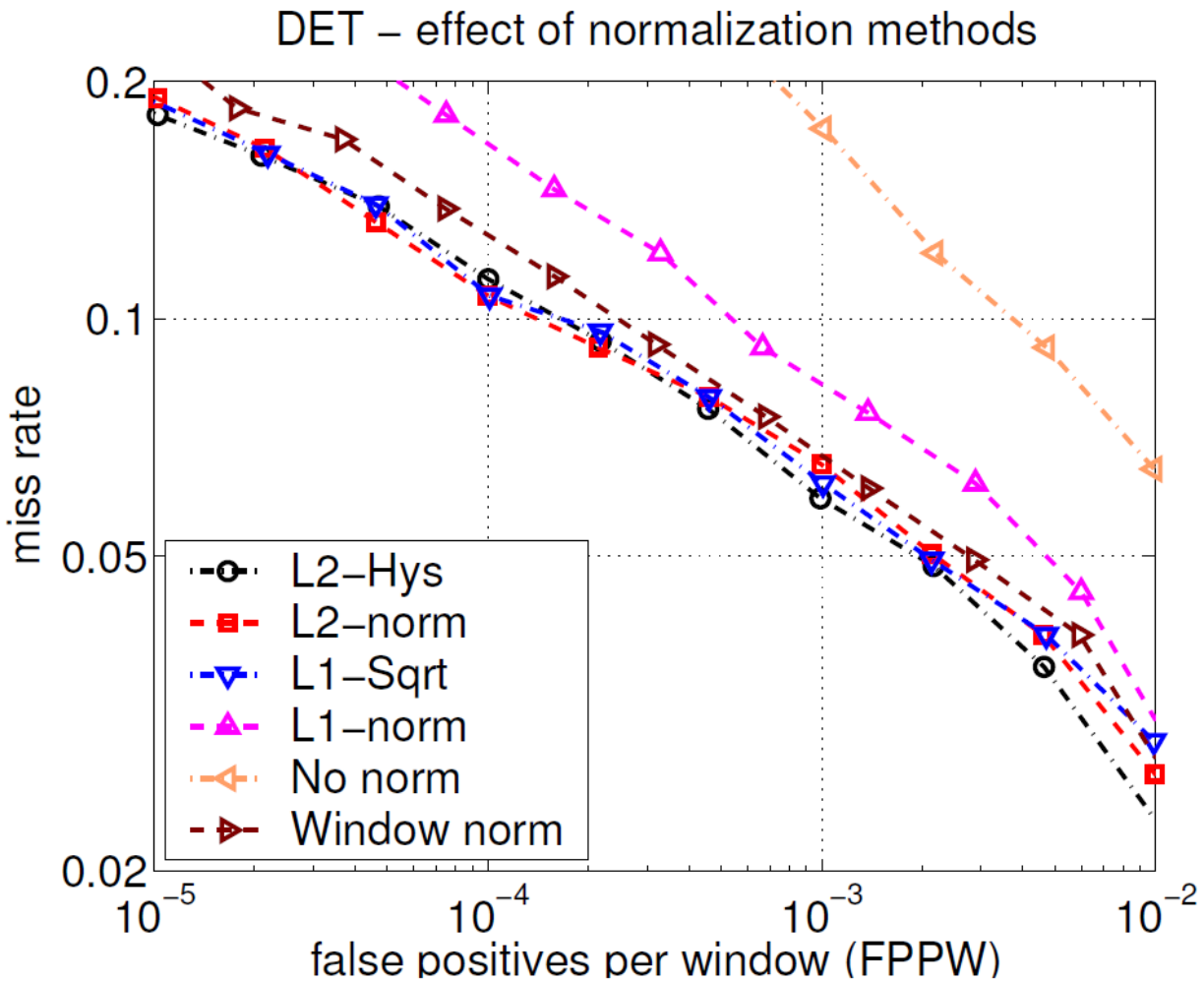
For 64x128 pixels, there are 7 blocks and 15 blocks in horizontal and vertical direction respectively

Total size of feature vector:
7*15*36 = 3780

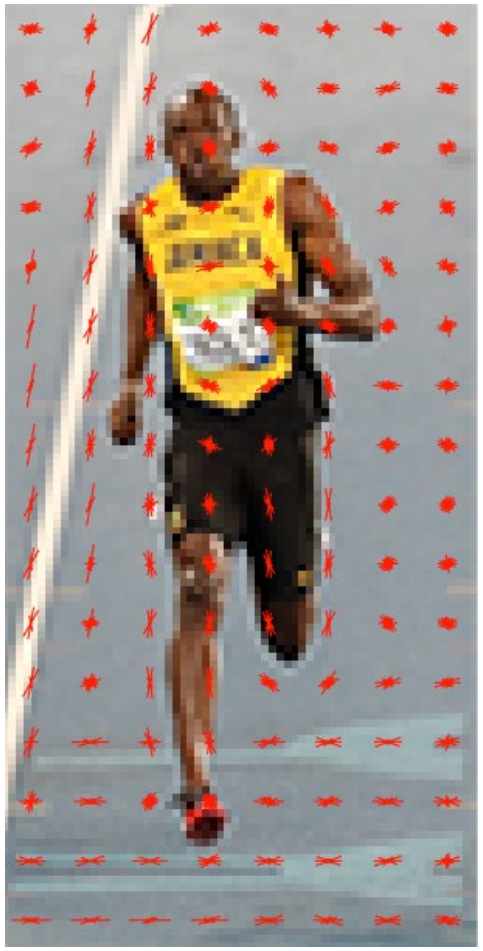
Effect of Overlapping



Effect of Normalization

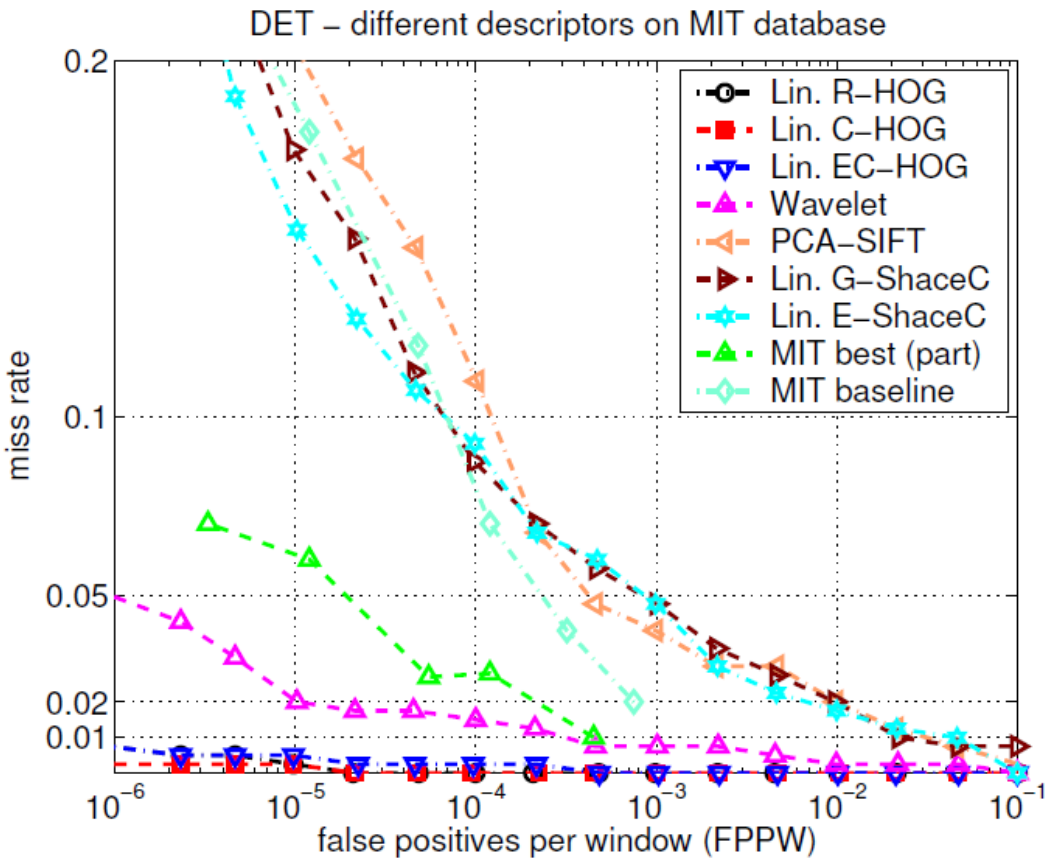


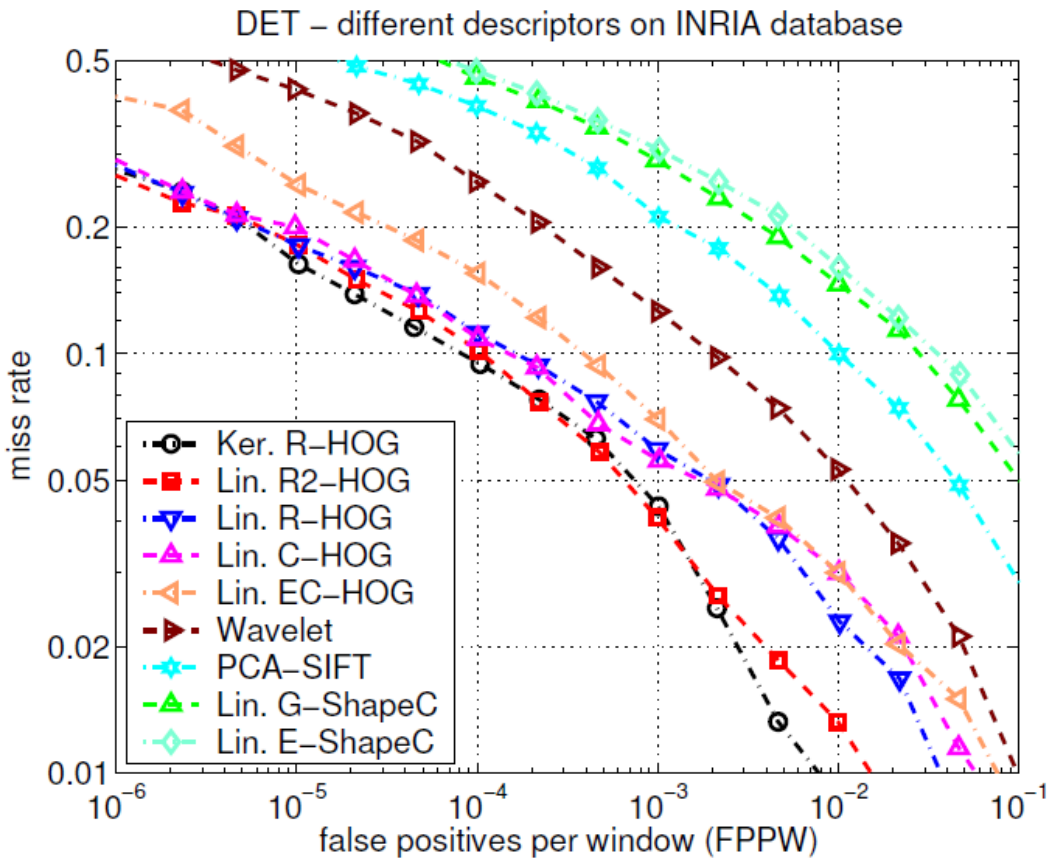
Visualizing Histogram of Oriented Gradients



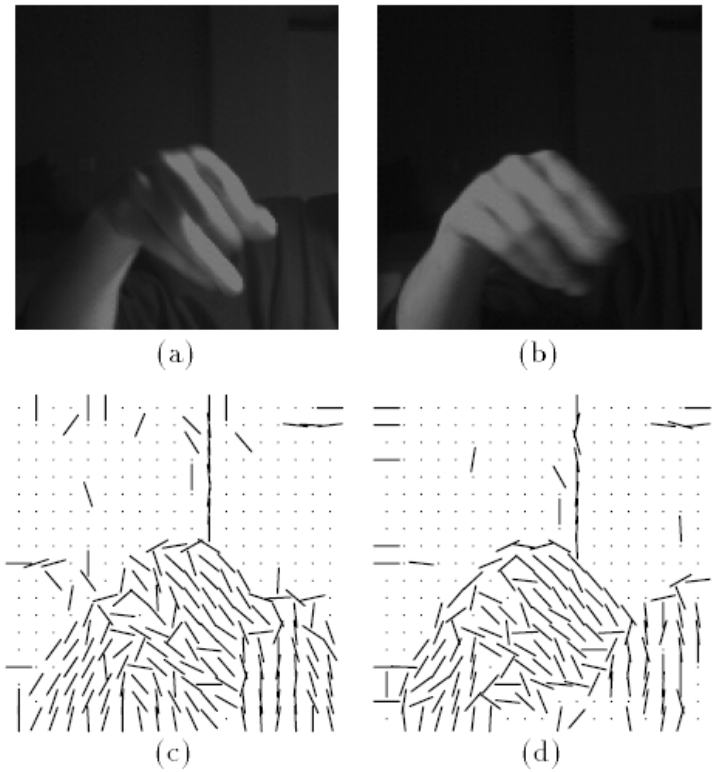
Applications of HOG – Pedestrian Detection

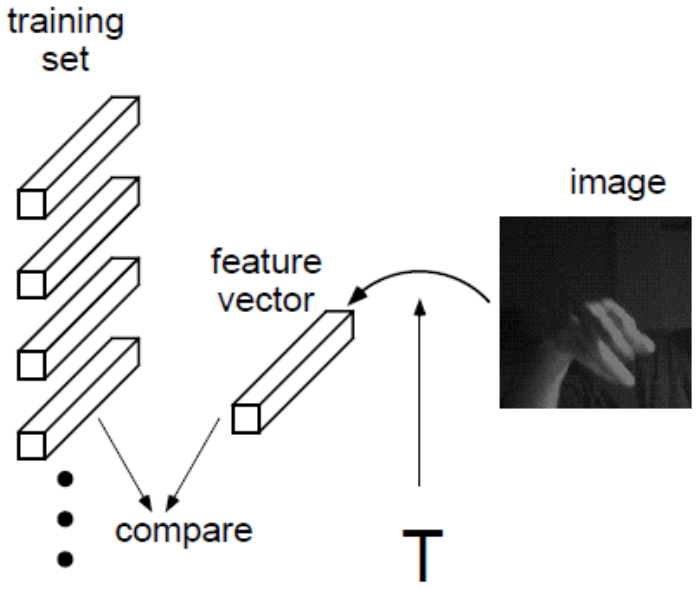
Result of HOG detectors





Application – Hand Gesture Recognition

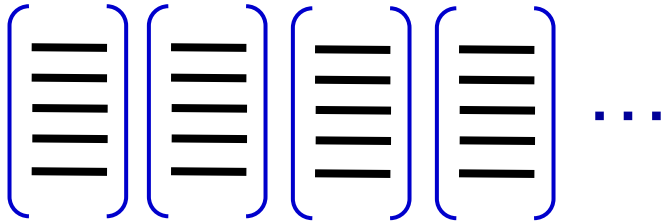




William T. Freeman, Michal Roth, **"Orientation Histograms for Hand Gesture Recognition"**, Tech. Rep. TR94-03, Mitsubishi Electric Research Laboratories, Cambridge, MA, December 1994.

Summary

- All SIFT, SURF, ORB and HOG describe the patches using feature vectors.



- Feature vectors can be used for image correspondence matching, such as stereo image correspondence or image stitching.

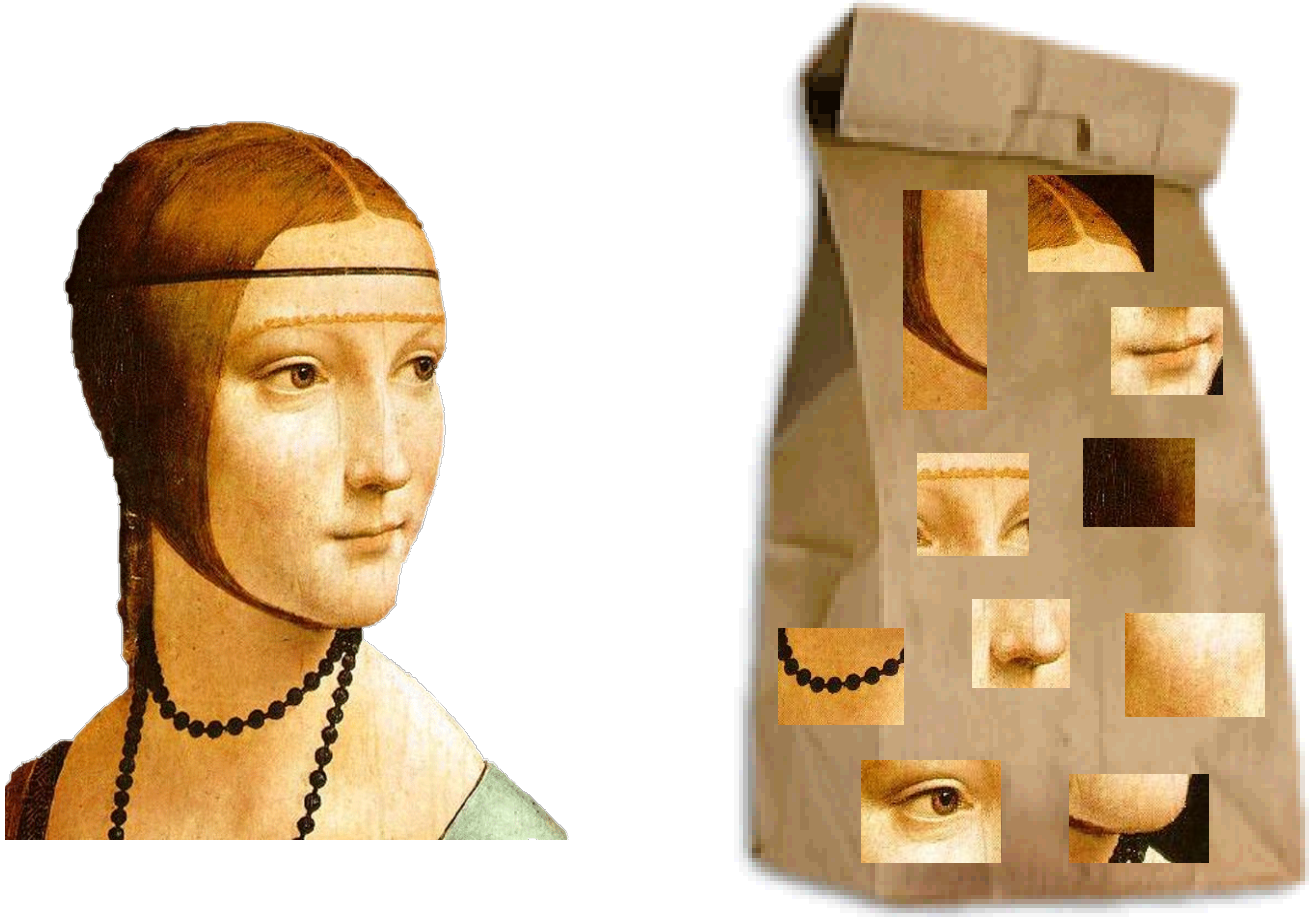
- Could we use the feature vectors for image retrieval? How?

Today's Agenda

- Shape Descriptor
  - Scale Invariant Feature Transform (SIFT)
  - Speed Up Robust Feature (SURF)
  - Oriented FAST and Rotated BRIEF (ORB)
  - Histogram of Oriented Graph (HOG)

- Visual Bags of Words.

Visual Bag-of-words



Many slides adapted from Fei-Fei Li, Rob Fergus, and Antonio Torralba

What is Visual Bag of Words

- **Definition:**
- Representation of a ***image feature*** as ***words***

- ***Goal:***

Given a ***compact representation*** of image

Enable searching of ***similar images*** in database.

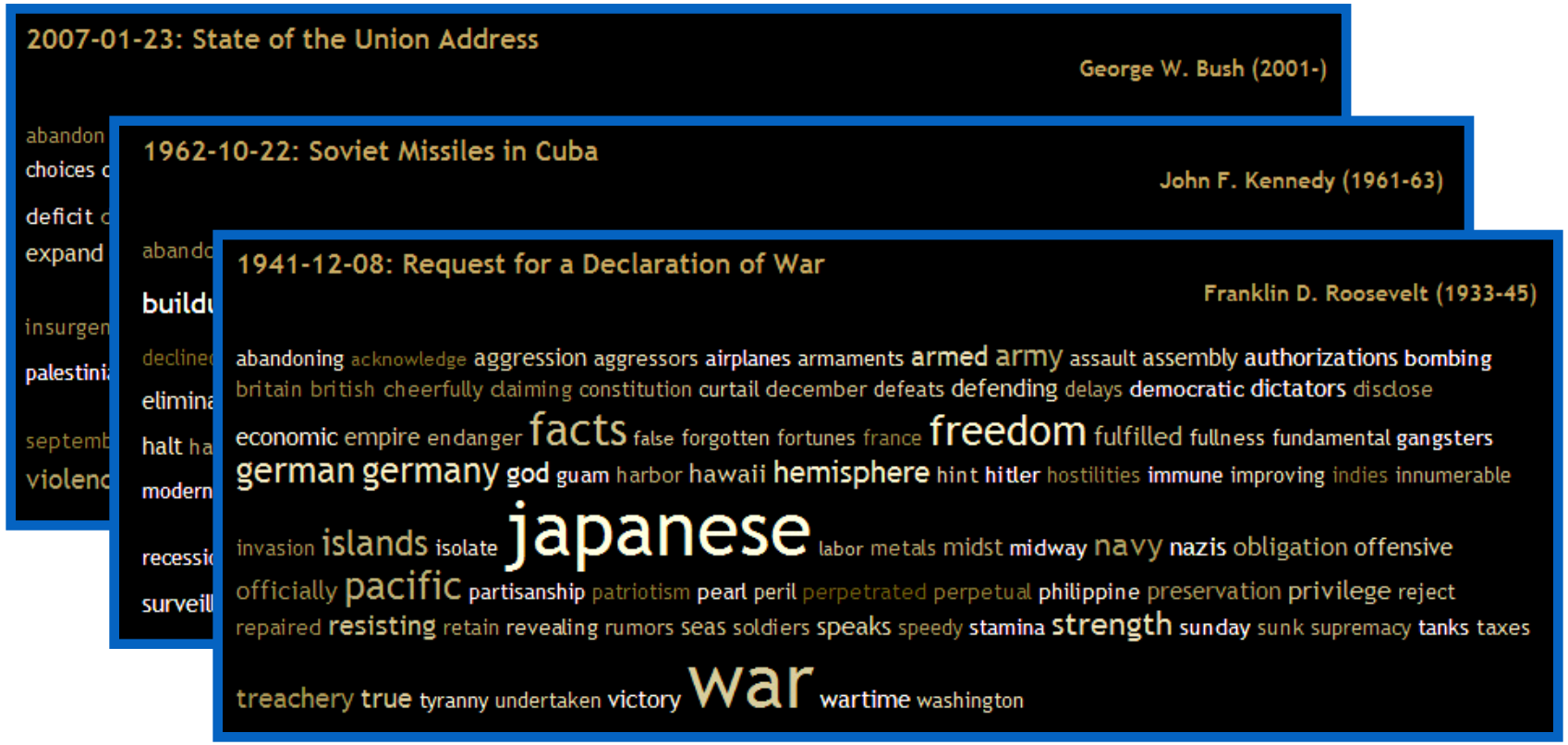
Archive ***image classifications***

Analogy of documents

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

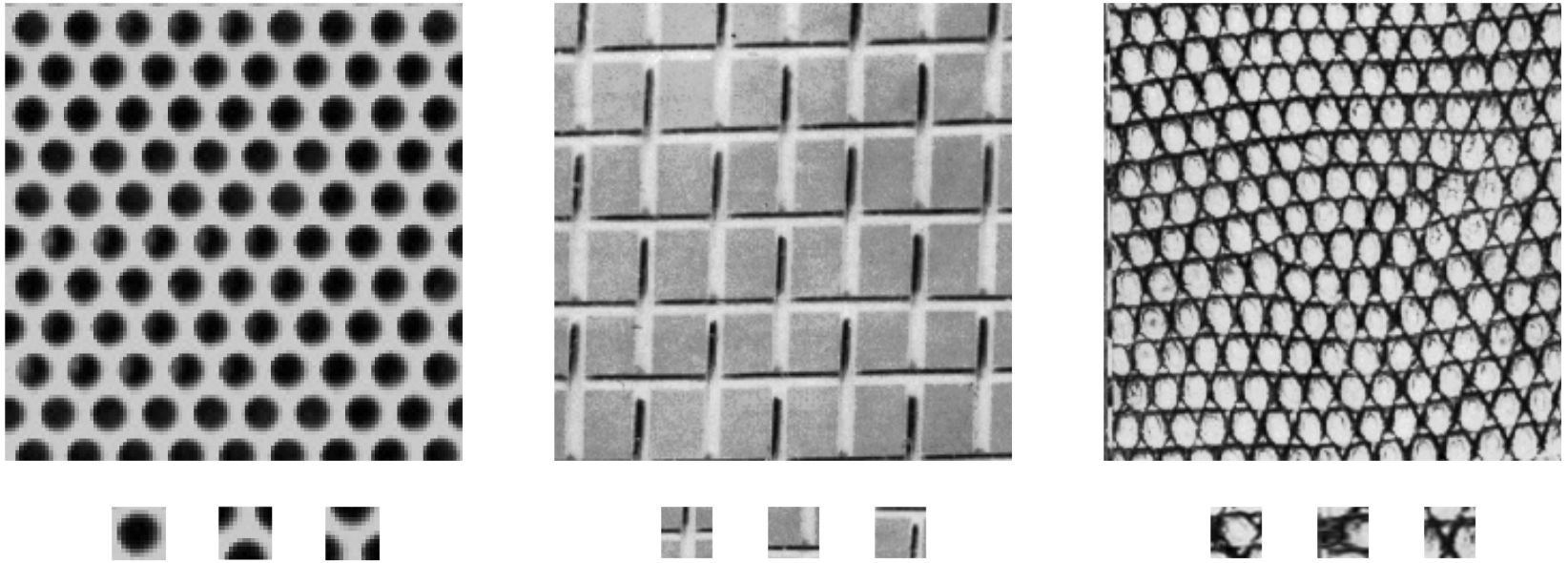
Analogy of Visual Bag of Words

Analogy of Visual Bag of Words

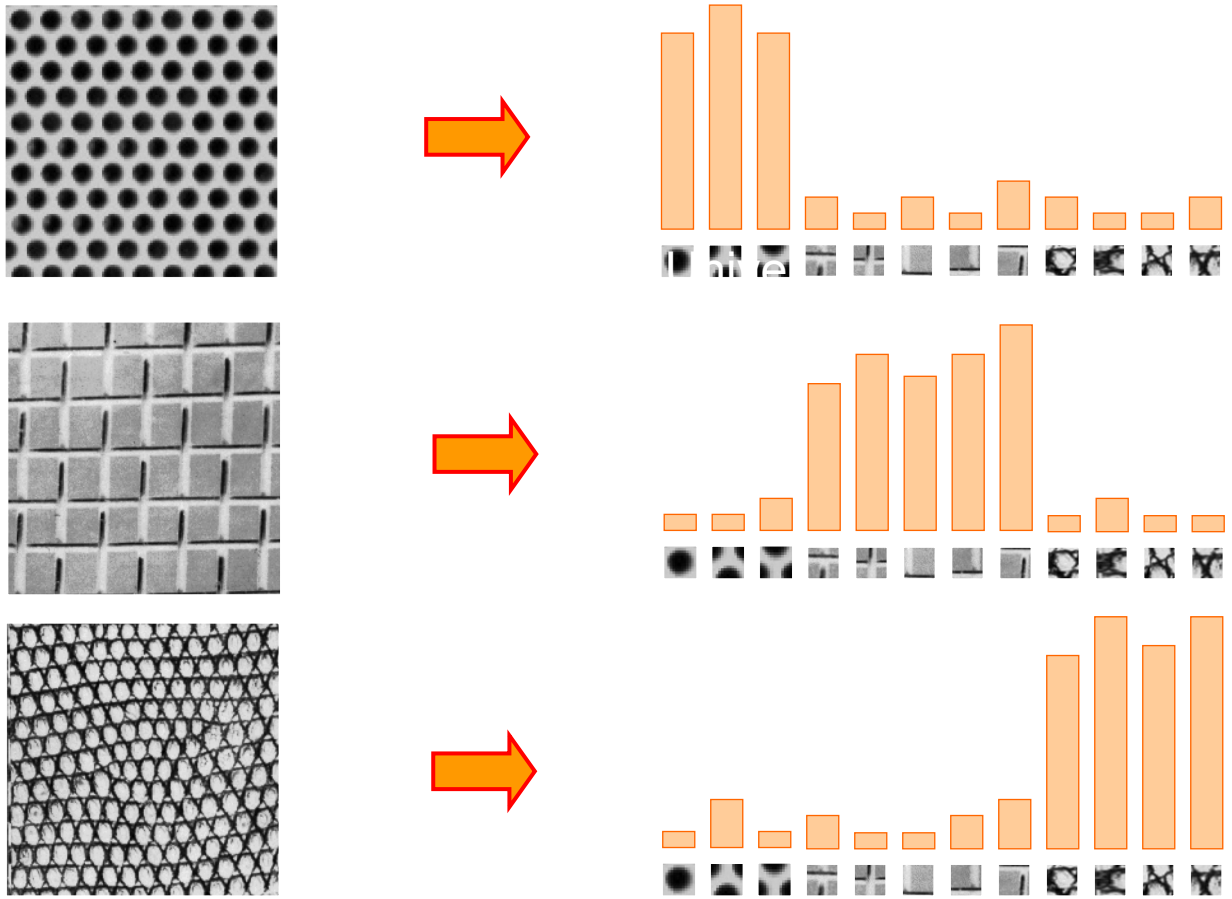


Origin of Visual Bag of Words – Texture Recognition

- Texture is characterized by the repetition of basic elements or ***textons***
- For stochastic textures, it is the identity of the ***textons***, ***NOT*** their ***spatial arrangement***, that matters

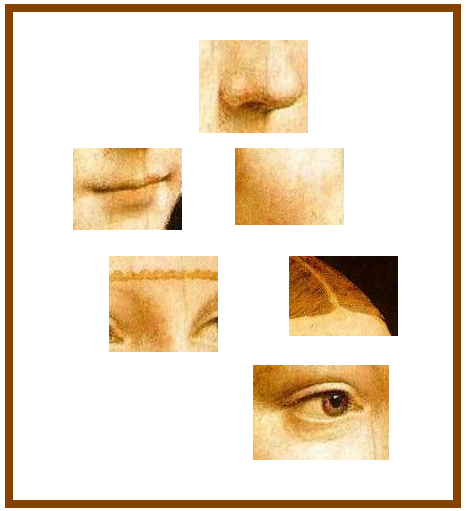


Texture Recognition



Bags of features for image classification

1. Extract features



Bags of features for image classification

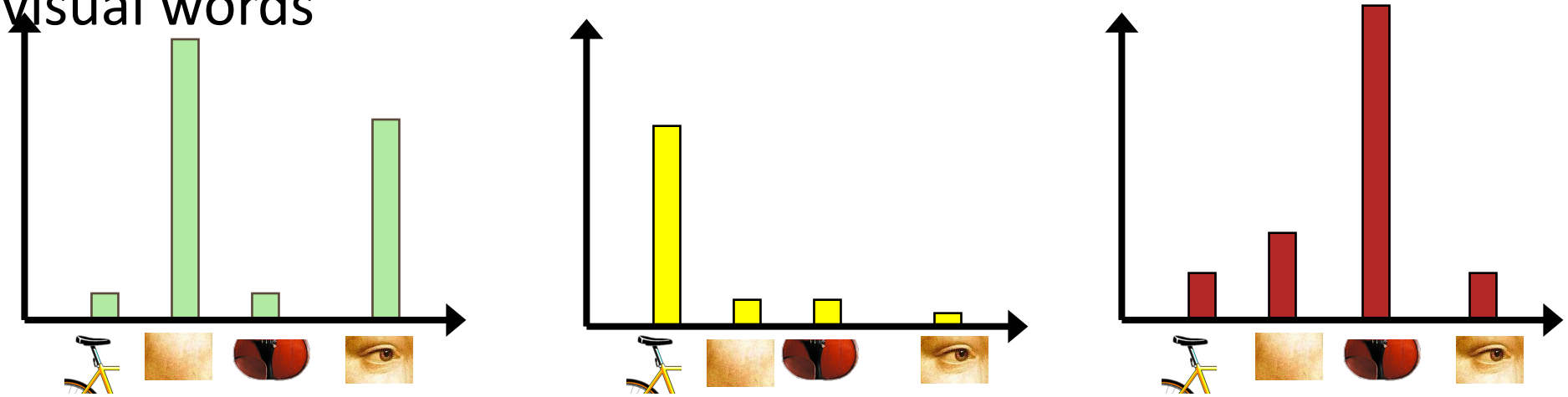
1. Extract features
2. Learn “visual vocabulary”

Bags of features for image classification

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary

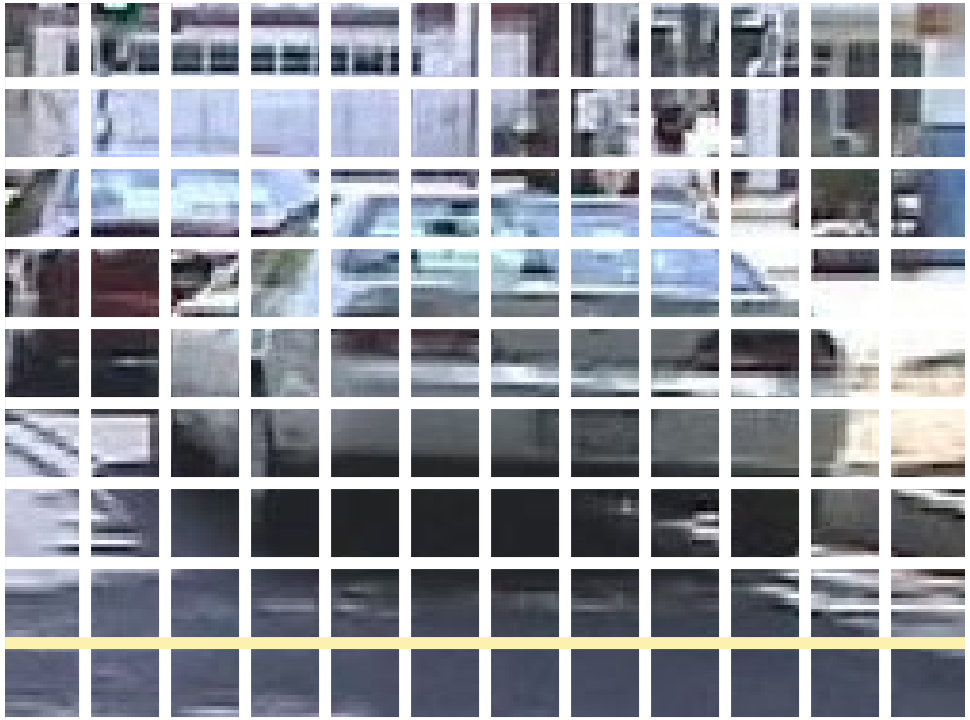
Bags of features for image classification

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”



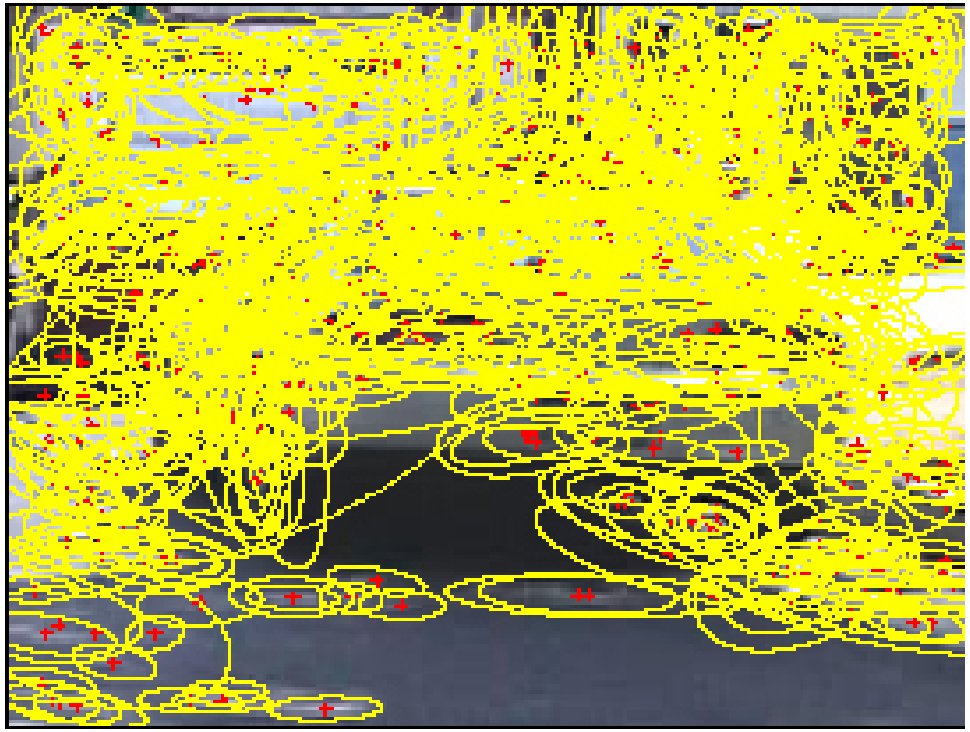
1. Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005



1. Feature extraction

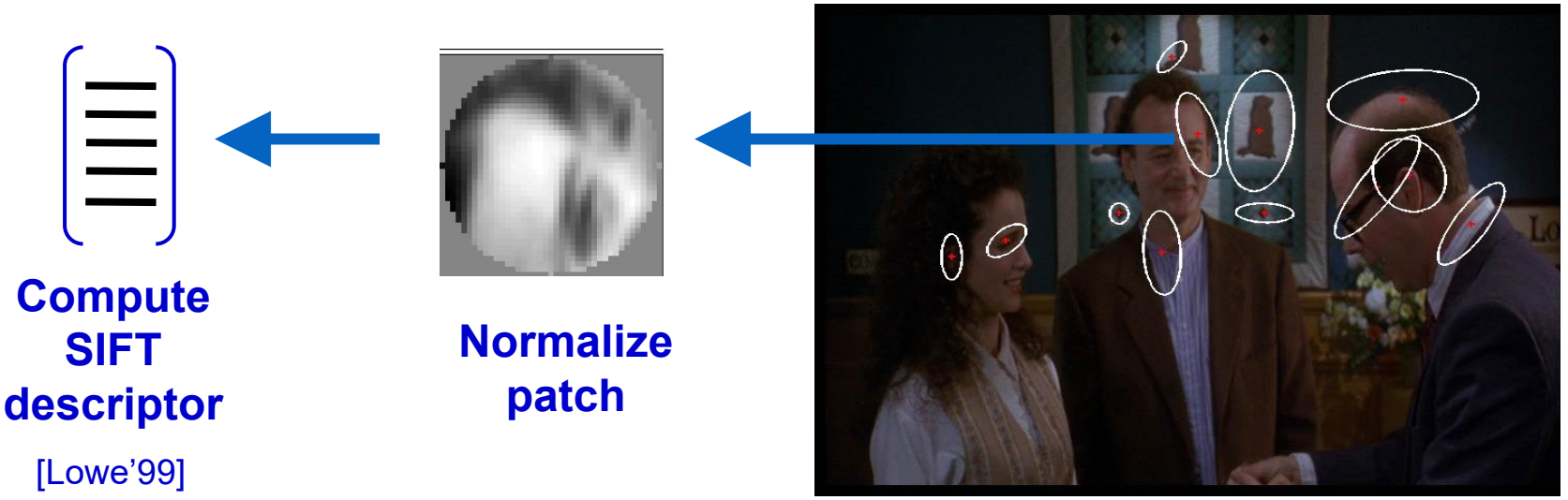
- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005



1. Feature extraction

- Regular grid
  - Vogel & Schiele, 2003
  - Fei-Fei & Perona, 2005
- Interest point detector
  - Csurka et al. 2004
  - Fei-Fei & Perona, 2005
  - Sivic et al. 2005
- Other methods
  - Random sampling (Vidal-Naquet & Ullman, 2002)
  - Segmentation-based patches (Barnard et al. 2003)

1. Feature extraction



Compute SIFT descriptor

[Lowe'99]

Normalize patch

Detect patches

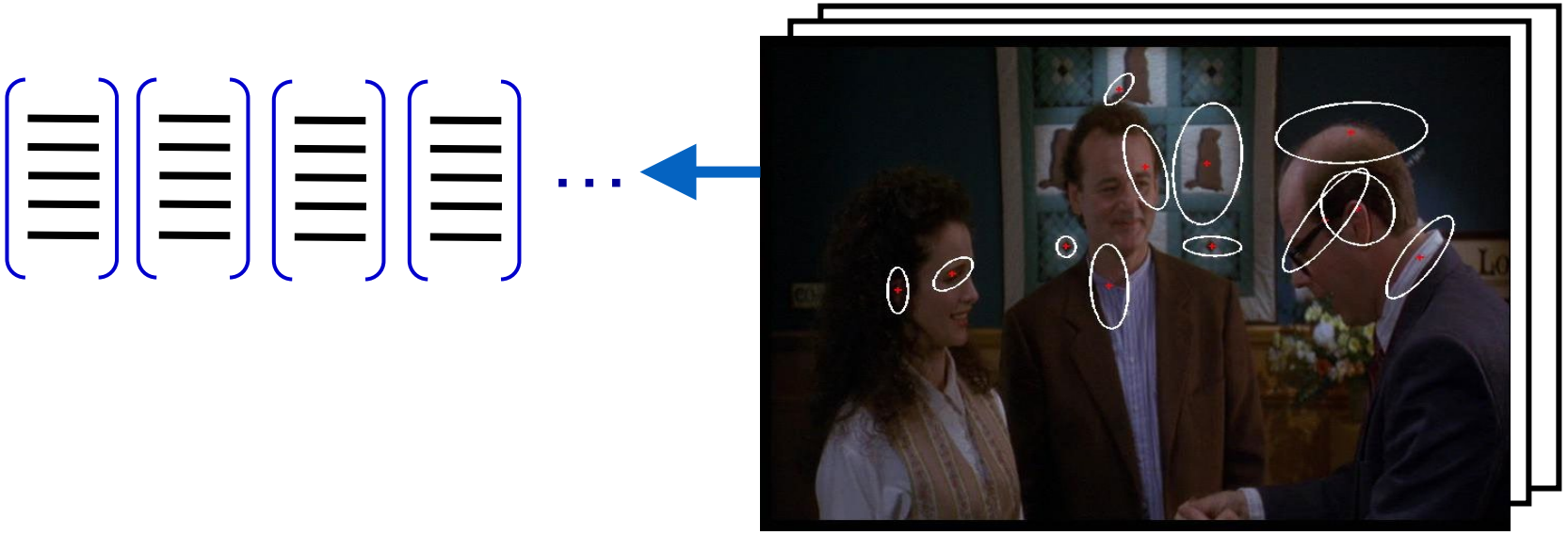
[Mikojaczyk and Schmid '02]

[Mata, Chum, Urban & Pajdla, '02]

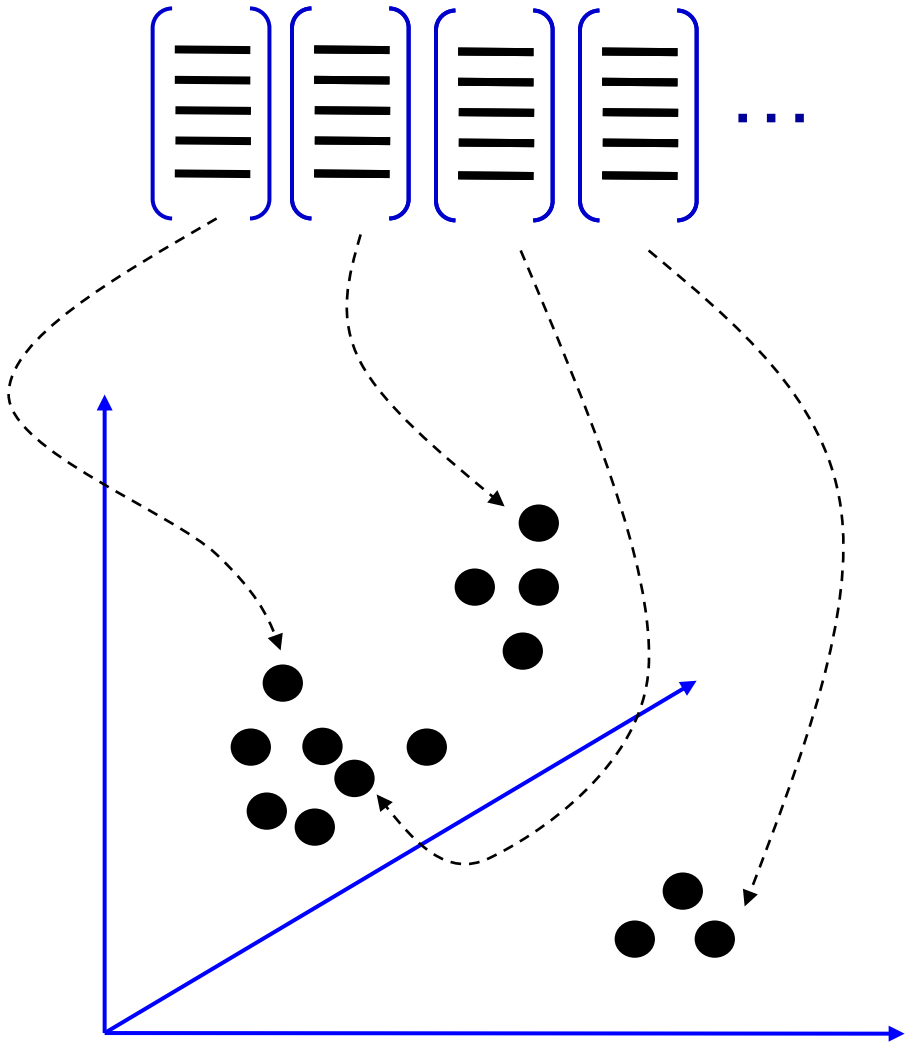
[Sivic & Zisserman, '03]

Slide credit: Josef Sivic

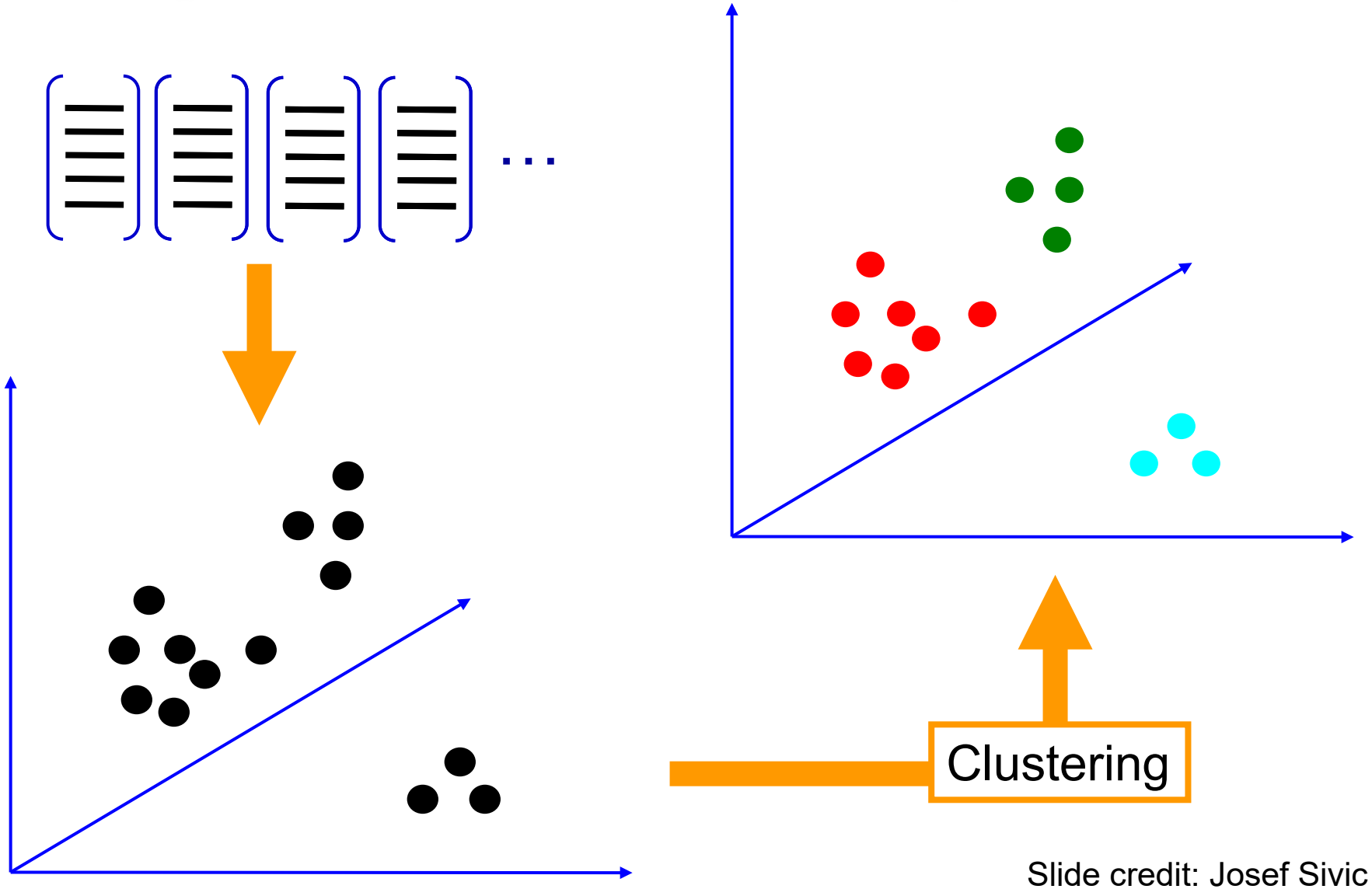
1. Feature extraction



2. Learning the visual vocabulary

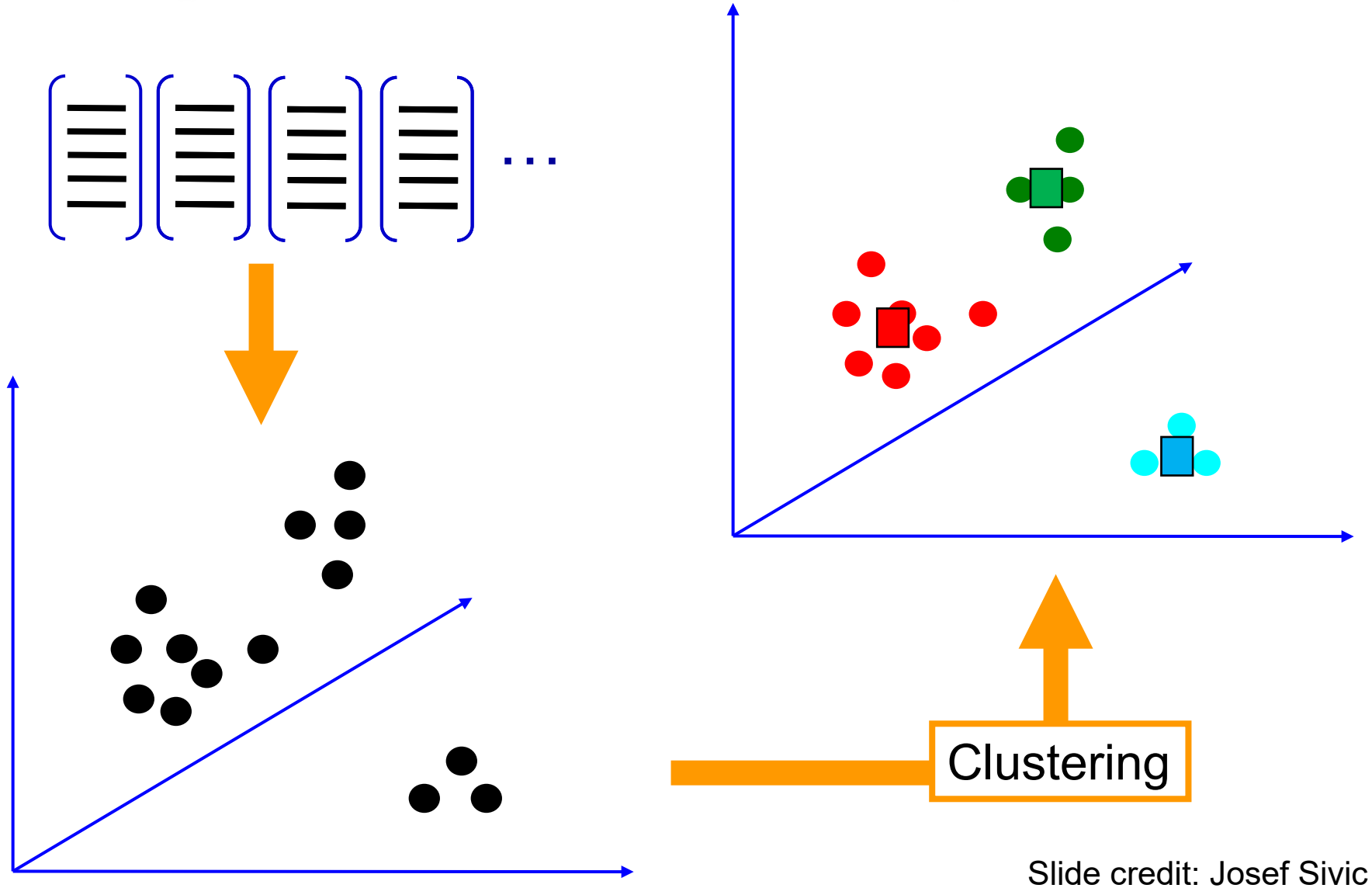


2. Learning the visual vocabulary



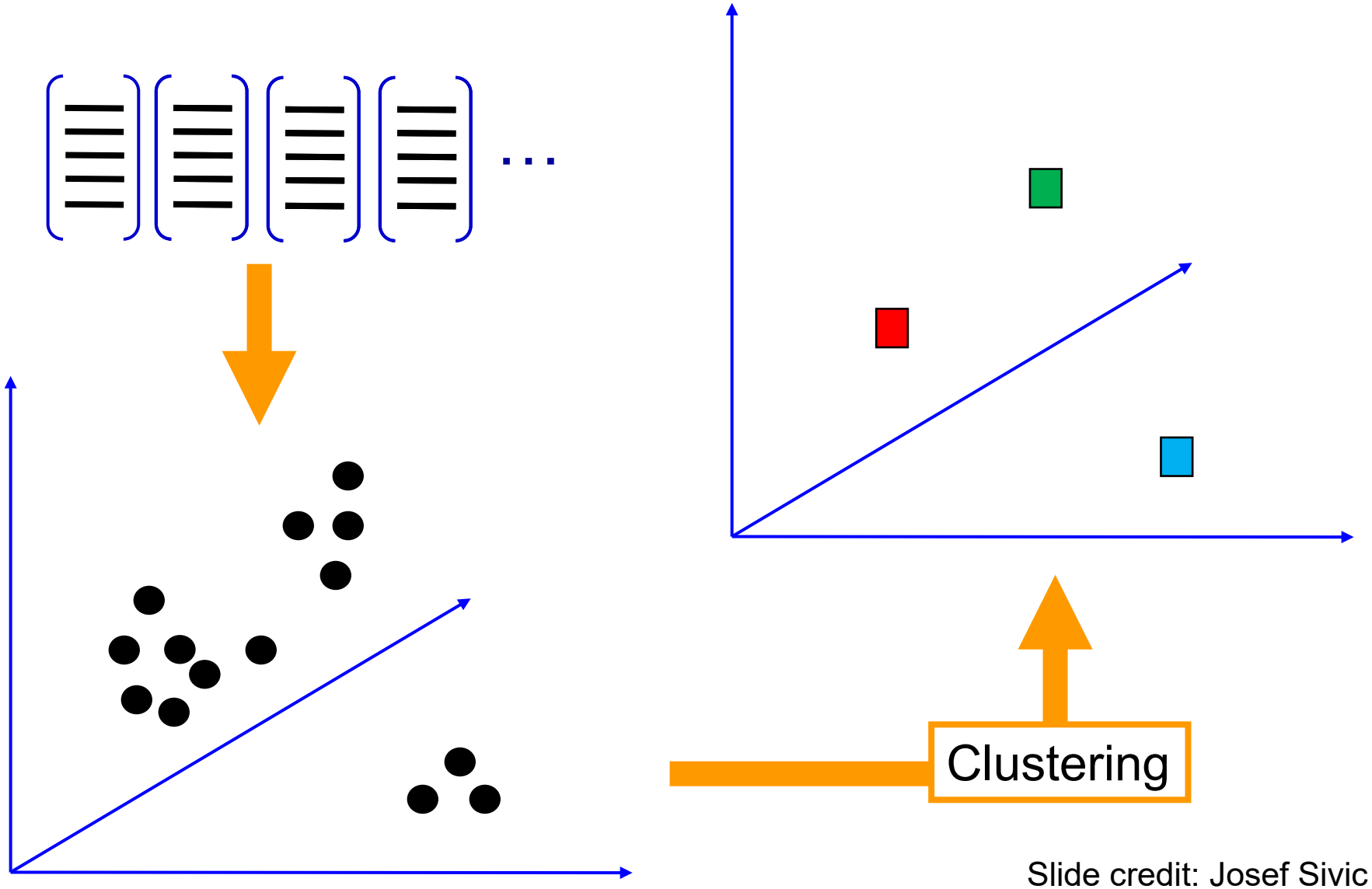
Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

2. Learning the visual vocabulary



Slide credit: Josef Sivic

K-means clustering

- Want to minimize sum of squared Euclidean distances between points x_i and their nearest cluster centers m_k

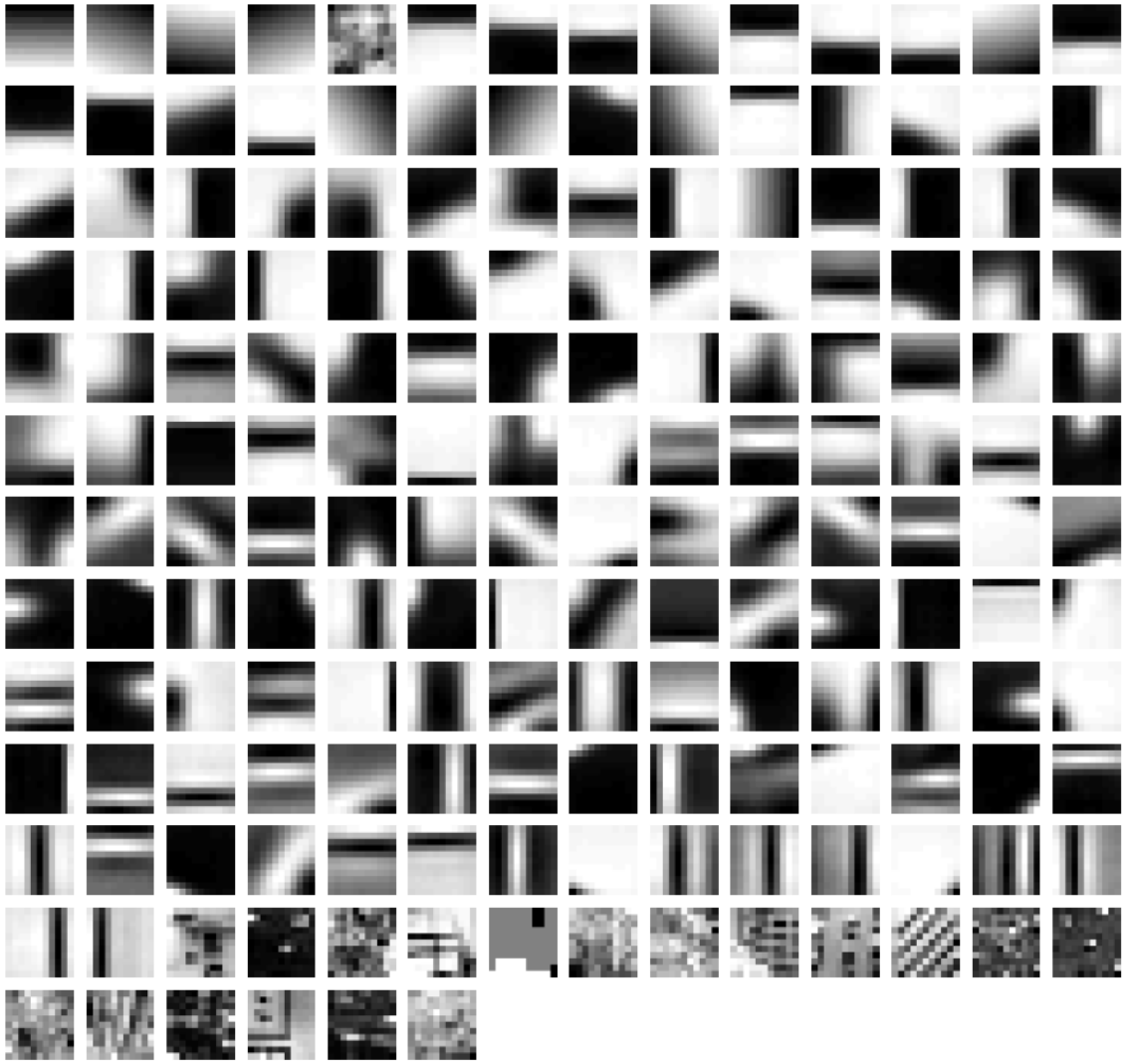
$$D(X,M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (x_i - m_k)^2$$

- Algorithm:
- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it

From clustering to vector quantization

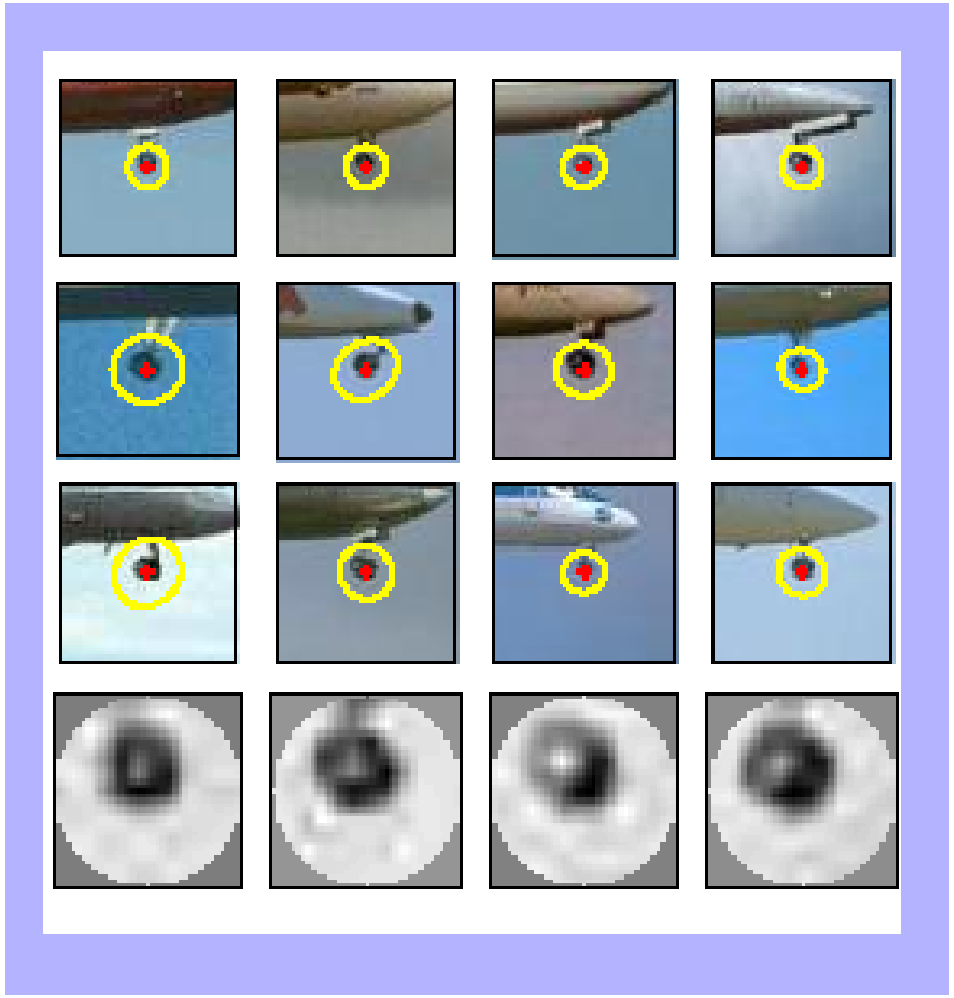
- Clustering is a common method for learning a visual vocabulary or codebook
  - Unsupervised learning process
  - Each cluster center produced by k-means becomes a codevector
  - Codebook can be learned on separate training set
  - Provided the training set is sufficiently representative, the codebook will be “universal”
- The codebook is used for quantizing features
  - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

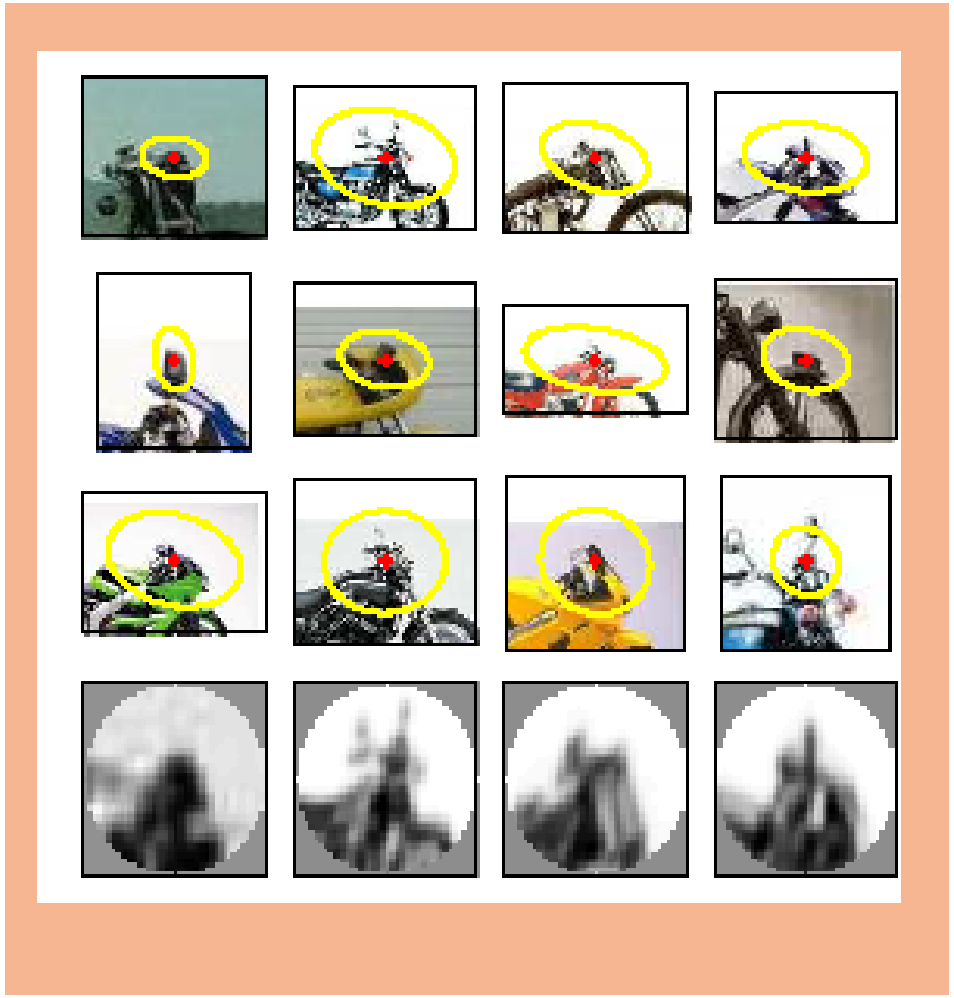
Example visual vocabulary



Fei-Fei et al. 2005

Image patch examples of visual words (Codebook)





Sivic et al. 2005

3. Image representation

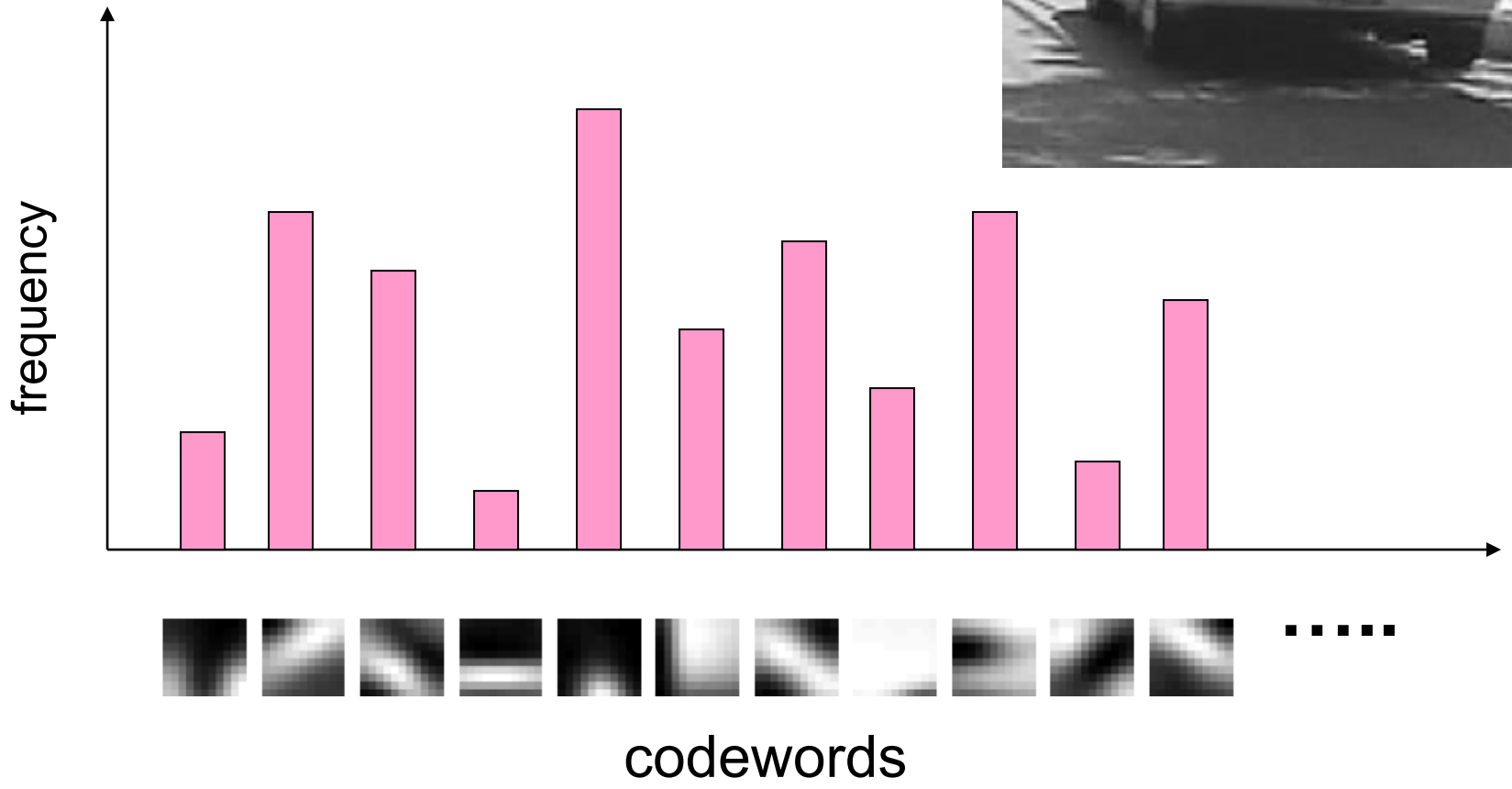
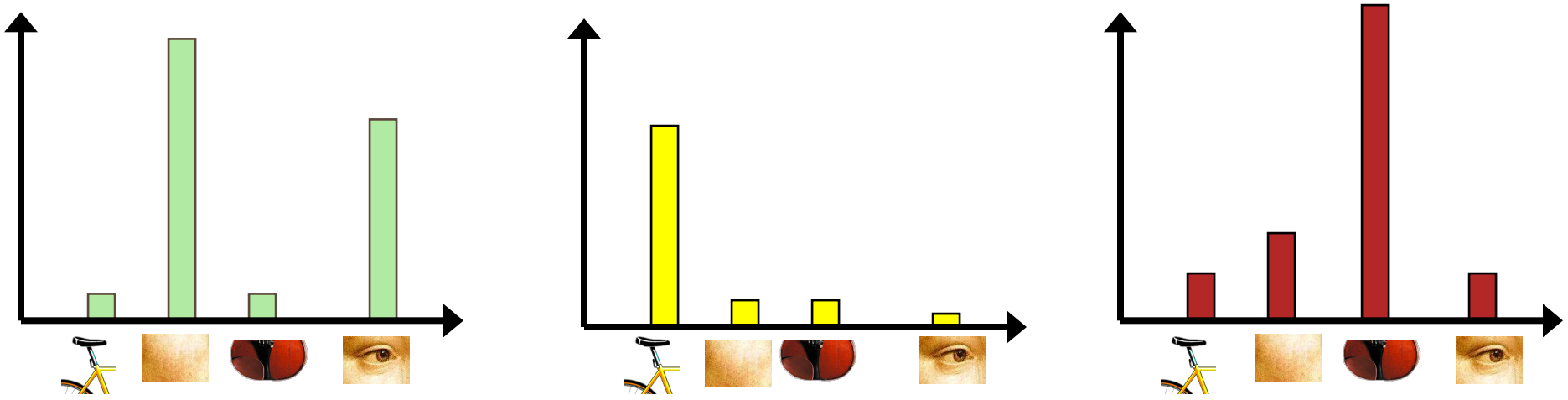


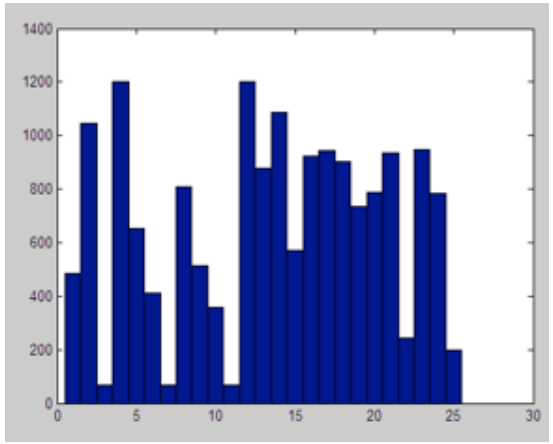
Image classification

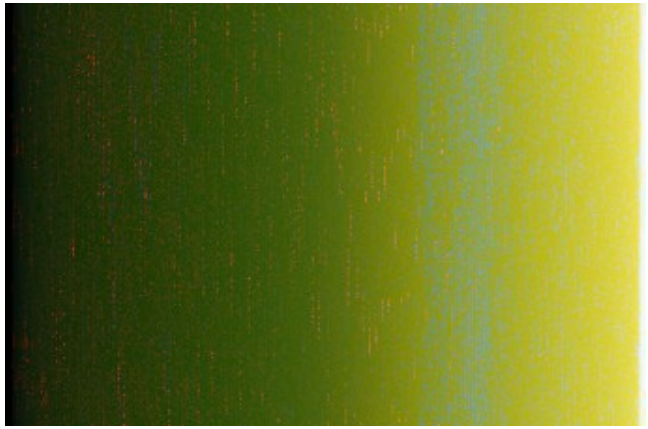
- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



Possible Extension to bag-of-words models

Spatial relationships of visual words



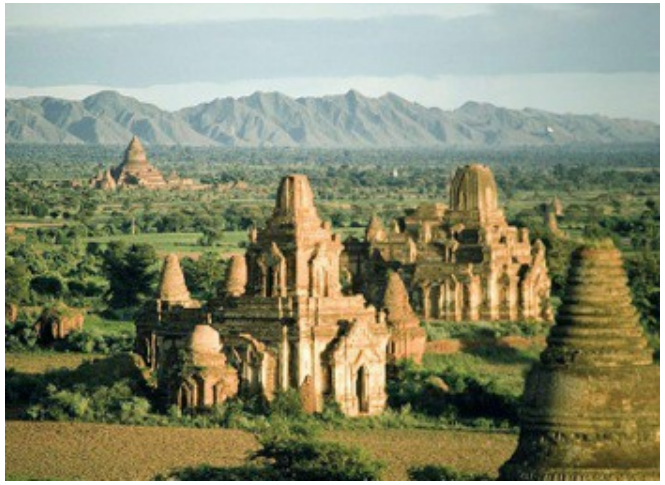


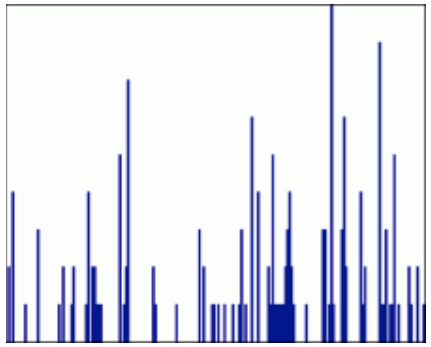
All of these images have the same color histogram

Slide Credit: Kris Kitani

Spatial pyramid representation

- Extension of a bag of features
- Locally orderless representation at several levels of resolution



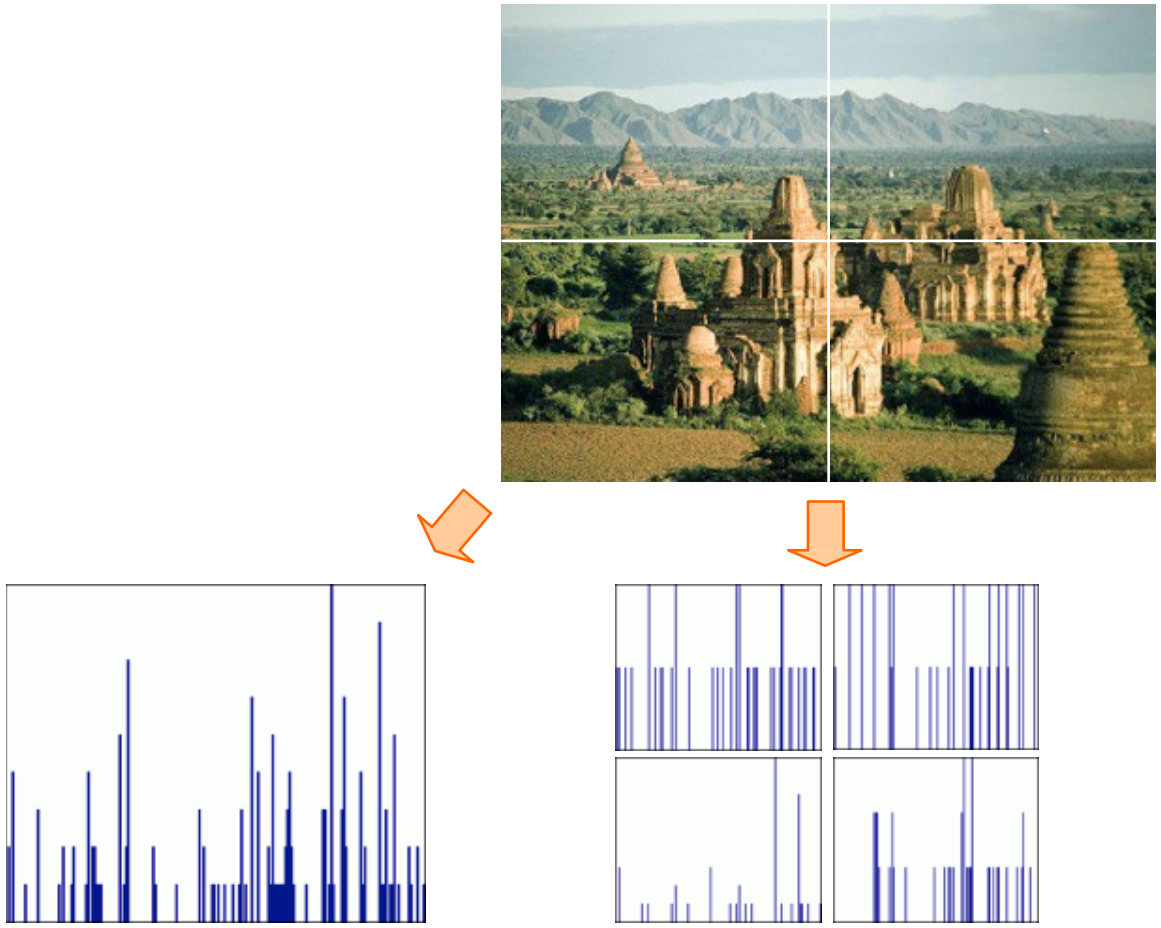


Lazebnik, Schmid & Ponce (CVPR 2006)

Slide Credit: Kris Kitani

Spatial pyramid representation

- Extension of a bag of features
- Locally orderless representation at several levels of resolution

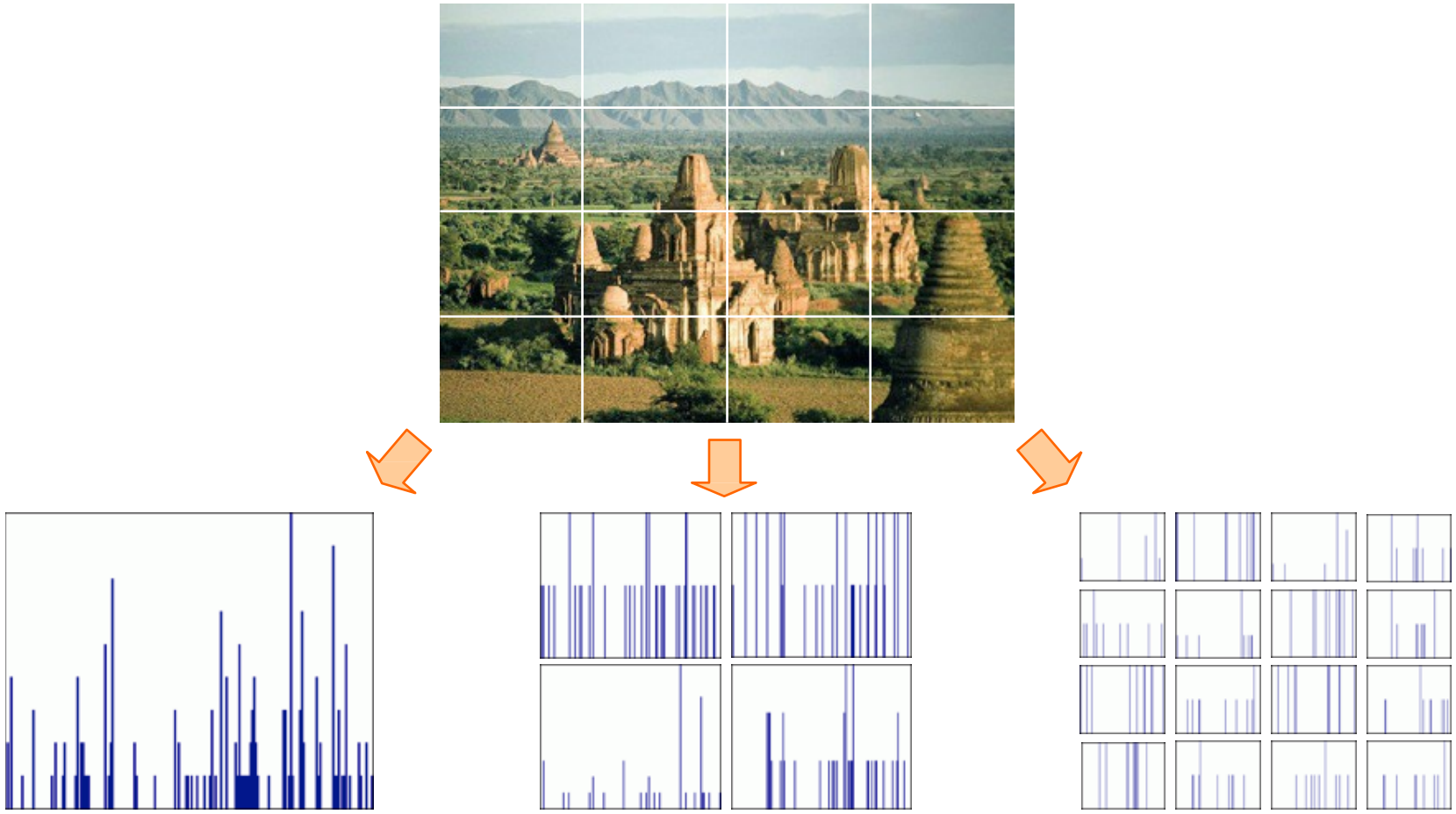


Lazebnik, Schmid & Ponce (CVPR 2006)

Slide Credit: Kris Kitani

Spatial pyramid representation

- Extension of a bag of features
- Locally orderless representation at several levels of resolution



Lazebnik, Schmid & Ponce (CVPR 2006)

Slide Credit: Kris Kitani

Summary of Visual Bag of Words

- **Advantages:**
  - largely unaffected by position and orientation of object in image
  - fixed length vector irrespective of number of detections
  - Very successful in classifying images according to the objects they contain
  - Still requires further testing for large changes in scale and viewpoint
- **Disadvantages:**
  - No explicit use of configuration of visual word positions
  - Poor at localizing objects within an image
  - Multi-resolution representation is possible to solve the issue.

Appendix

- Edge Detection in MATLAB
- HARRIS in MATLAB
- SIFT in MATLAB
- SURF in MATLAB

Harris Corner Detector in MATLAB

```
points = detectHarrisFeatures(I)
points = detectHarrisFeatures(I,Name,Value)
```

Description

`points = detectHarrisFeatures(I)` returns a `cornerPoints` object, `points`. The object contains information about the feature points detected in a 2-D input image, `I`. The `detectHarrisFeatures` function uses the Harris–Stephens algorithm to find these feature points.

`points = detectHarrisFeatures(I,Name,Value)` uses additional options specified by one or more `Name,Value` pair arguments.

https://www.mathworks.com/help/vision/ref/detectharrisfeatures.htm

SIFT Software

- MATLAB code
  - -http://www.vlfeat.org
  - -Download and put into directory (say: c:\Program Files\vlfeatroot)
  - At MATLAB prompt, type
  - run('c:\Program Files\vlfeatroot\toolbox\vl_setup');
- Main functions
  - vl_sift- extract SIFT features from an image
  - vl_ubcmatch – match two sets of SIFT features

Extracting SIFT features

- Function call

  [f, d]=vl_sift(I)

- Returns

  Arrays f(4, N), d(128,N), where N is the number of features

  d(1:128,i) is the 128 element descriptor of i^{th} feature

  f(1:4,i) is (x,y,σ,angle) for i^{th} feature

MATLAB Code

SIFT.m

```
%Read the image
I1 = imread('D:\data\box.png');
I1 = single(I1); %convert the image to float
imshow(I1, []); %show the image

peak_thresh = 0 %threshold you can play around
edge_thresh = 10
[f1,d1]=vl_sift(I1,'PeakThresh', peak_thresh, 'EdgeThresh', edge_thresh
fprintf('Number of feature detected: %d\n', size(f1,2));
%detecting SIFT feature

h=vl_plotframe(f1); %plot Sift Feature
set(h, 'color', 'r', 'linewidth', 1);
```

Matching Features

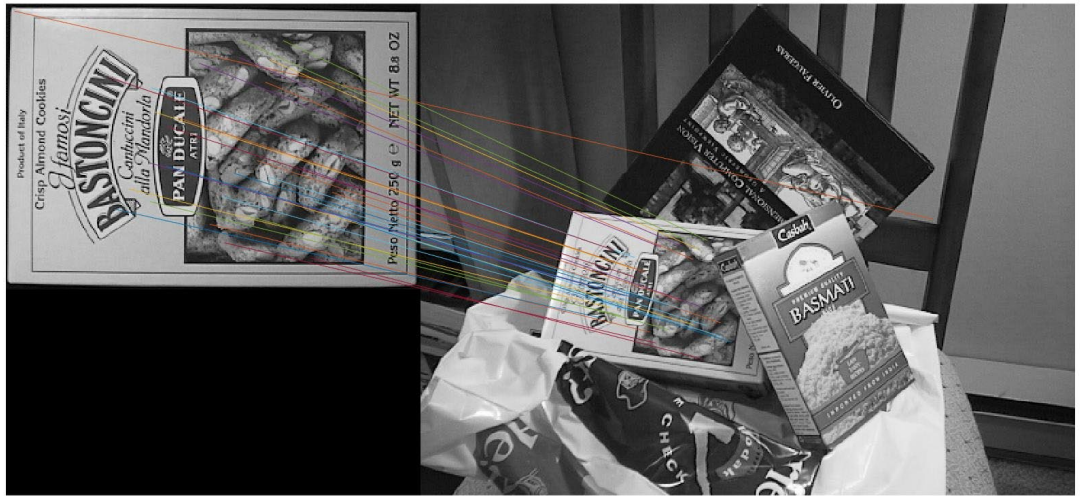
- Function Call
  - [matches, scores] = vl_ubcmatch(d1, d2);
- Returns
  - Arrays: matches(2, M), scores(M), where M is number of matches
  - Matches(1:2, i) are the indices of features for ith match
  - Scores(i) is the squared Euclidean distance between the features

```
%matching features
thresh = 3.0;
[matches, scores] = vl_ubcmatch(d1, d2, thresh);
fprintf('Number of matching frames (features): %d\n', size(matches, 2));

indices1 = matches(1,:);
f1match = f1(:, indices1);
d1match = d1(:, indices1);

indices2 = matches(2,:);
f2match = f2(:, indices2);
d2match = d2(:, indices2);
%Displaying matched features
[i1,j1]=ndgrid(1:size(I1,1), 1:size(I1,2));
[i2,j2]=ndgrid(1:size(I2,1), (1:size(I2,2))+size(I1,2));
I3=accumarray([i1(:),j1(:);i2(:),j2(:)],[I1(:);I2(:)]);

imshow(I3, []); %show the image
s=single(size(I1,2));
line([f1match(1,:);f2match(1,:)+s], [f1match(2,:);f2match(2,:)]);
```



SURF in MATLAB

- MATLAB does not provide SIFT function, you may refer to a similar function SURF

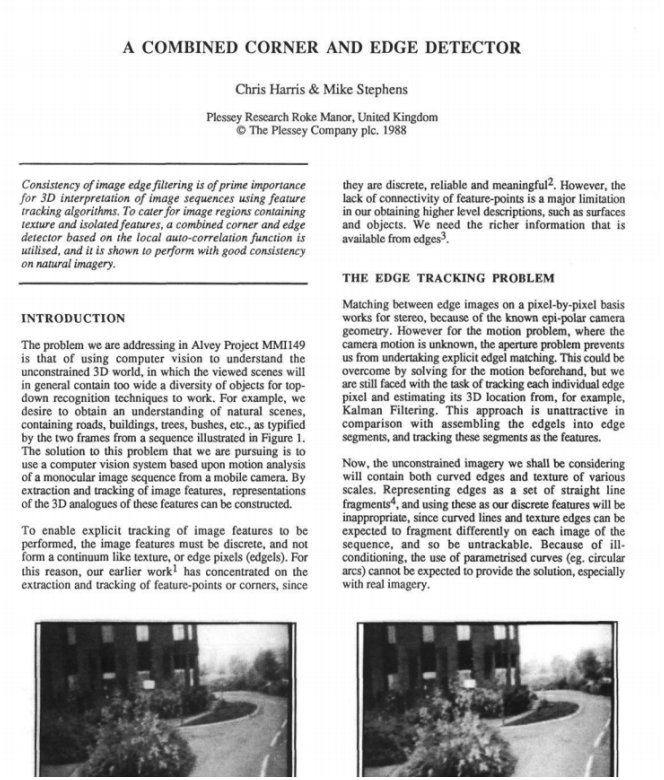
```
I = imread('cameraman.tif');
points = detectSURFFeatures(I);
```

Display locations of interest in image.

```
imshow(I); hold on;
plot(points.selectStrongest(10));
```

https://www.mathworks.com/help/vision/ref/detectsurffeatures.html

Reference



A COMBINED CORNER AND EDGE DETECTOR

Chris Harris & Mike Stephens

Plessey Research Roke Manor, United Kingdom
© The Plessey Company plc. 1988

Consistency of image edge filtering is of prime importance for 3D interpretation of image sequences using feature tracking algorithms. To cater for image regions containing texture and isolated features, a combined corner and edge detector based on the local auto-correlation function is utilised, and it is shown to perform with good consistency on natural imagery.

INTRODUCTION

The problem we are addressing in Alvey Project MMI149 is that of using computer vision to understand the unconstrained 3D world, in which the viewed scenes will in general contain too wide a diversity of objects for top-down recognition techniques to work. For example, we desire to obtain an understanding of natural scenes, containing roads, buildings, trees, bushes, etc., as typified by the two frames from a sequence illustrated in Figure 1. The solution to this problem that we are pursuing is to use a computer vision system based upon motion analysis of a monocular image sequence from a mobile camera. By extraction and tracking of image features, representations of the 3D analogues of these features can be constructed.

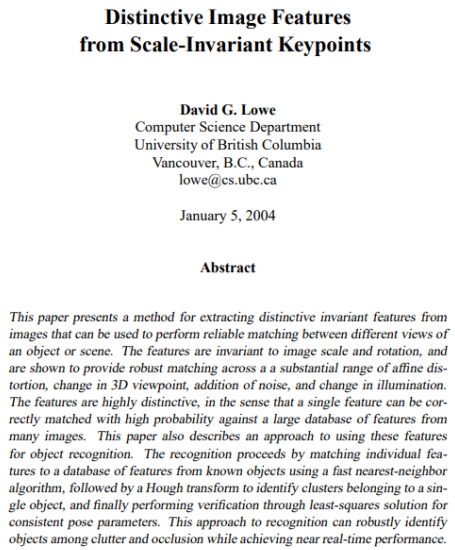
To enable explicit tracking of image features to be performed, the image features must be discrete, and not form a continuum like texture, or edge pixels (edgels). For this reason, our earlier work[1] has concentrated on the extraction and tracking of feature-points or corners, since they are discrete, reliable and meaningful[2]. However, the lack of connectivity of feature-points is a major limitation in our obtaining higher level descriptions, such as surfaces and objects. We need the richer information that is available from edges[3].

THE EDGE TRACKING PROBLEM

Matching between edge images on a pixel-by-pixel basis works for stereo, because of the known epi-polar camera geometry. However for the motion problem, where the camera motion is unknown, the aperture problem prevents us from undertaking explicit edgel matching. This could be overcome by solving for the motion beforehand, but we are still faced with the task of tracking each individual edge pixel and estimating its 3D location from, for example, Kalman Filtering. This approach is unattractive in comparison with assembling the edgels into edge segments, and tracking these segments as the features.

Now, the unconstrained imagery we shall be considering will contain both curved edges and texture of various scales. Representing edges as a set of straight line fragments[4], and using these as our discrete features will be inappropriate, since curved lines and texture edges can be expected to fragment differently on each image of the sequence, and so be untrackable. Because of ill-conditioning, the use of parametrised curves (eg. circular arcs) cannot be expected to provide the solution, especially with real imagery.

Chris Harris and Mike Stephens (1988). "A Combined Corner and Edge Detector". *Alvey Vision Conference*. **15**.



Distinctive Image Features from Scale-Invariant Keypoints

David G. Lowe
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
lowe@cs.ubc.ca

January 5, 2004

Abstract

This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

David G. Lowe, **"Distinctive image features from scale-invariant keypoints,"** *International Journal of Computer Vision,* 60, 2 (2004), pp. 91-110

Reference

SURF: Speeded Up Robust Features

Herbert Bay[1], Tinne Tuytelaars[2], and Luc Van Gool[12]

[1] ETH Zurich
{bay, vangool}@vision.ee.ethz.ch
[2] Katholieke Universiteit Leuven
{Tinne.Tuytelaars, Luc.Vangool}@esat.kuleuven.be

Abstract. In this paper, we present a novel scale- and rotation-invariant interest point detector and descriptor, coined SURF (Speeded Up Robust Features). It approximates or even outperforms previously proposed schemes with respect to repeatability, distinctiveness, and robustness, yet can be computed and compared much faster.
This is achieved by relying on integral images for image convolutions; by building on the strengths of the leading existing detectors and descriptors (in casu, using a Hessian matrix-based measure for the detector, and a distribution-based descriptor); and by simplifying these methods to the essential. This leads to a combination of novel detection, description, and matching steps. The paper presents experimental results on a standard evaluation set, as well as on imagery obtained in the context of a real-life object recognition application. Both show SURF's strong performance.

1 Introduction

The task of finding correspondences between two images of the same scene or object is part of many computer vision applications. Camera calibration, 3D reconstruction, image registration, and object recognition are just a few. The search for discrete image correspondences – the goal of this work – can be divided into three main steps. First, 'interest points' are selected at distinctive locations in the image, such as corners, blobs, and T-junctions. The most valuable property of an interest point *detector* is its repeatability, i.e. whether it reliably finds the same interest points under different viewing conditions. Next, the neighbourhood of every interest point is represented by a feature vector. This *descriptor* has to be distinctive and, at the same time, robust to noise, detection errors, and geometric and photometric deformations. Finally, the descriptor vectors are *matched* between different images. The matching is often based on a distance between the vectors, e.g. the Mahalanobis or Euclidean distance. The dimension of the descriptor has a direct impact on the time this takes, and a lower number of dimensions is therefore desirable.

It has been our goal to develop both a detector and descriptor, which in comparison to the state-of-the-art are faster to compute, while not sacrificing performance. In order to succeed, one has to strike a balance between the above

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In European Conference on Computer Vision, May 2006. 1, 2

ORB: an efficient alternative to SIFT or SURF

Ethan Rublee Vincent Rabaud Kurt Konolige Gary Bradski
Willow Garage, Menlo Park, California
{erublee}{vrabaud}{konolige}{bradski}@willowgarage.com

Abstract

Feature matching is at the base of many computer vision problems, such as object recognition or structure from motion. Current methods rely on costly descriptors for detection and matching. In this paper, we propose a very fast binary descriptor based on BRIEF*, called* ORB*, which is rotation invariant and resistant to noise. We demonstrate through experiments how* ORB *is at two orders of magnitude faster than* SIFT*, while performing as well in many situations. The efficiency is tested on several real-world applications, including object detection and patch-tracking on a smart phone.*

1. Introduction

The SIFT keypoint detector and descriptor [17], although over a decade old, have proven remarkably successful in a number of applications using visual features, including object recognition [17], image stitching [28], visual mapping [25], etc. However, it imposes a large computational burden, especially for real-time systems such as visual odometry, or for low-power devices such as cellphones. This has led to an intensive search for replacements with lower computation cost; arguably the best of these is SURF [2]. There has also been research aimed at speeding up the computation of SIFT, most notably with GPU devices [26].

In this paper, we propose a computationally-efficient replacement to SIFT that has similar matching performance, is less affected by image noise, and is capable of being used for real-time performance. Our main motivation is to enhance many common image-matching applications, e.g., to enable low-power devices without GPU acceleration to perform panorama stitching and patch tracking, and to reduce the time for feature-based object detection on standard PCs. Our descriptor performs as well as SIFT on these tasks (and

Figure 1. Typical matching result using ORB on real-world images with viewpoint change. Green lines are valid matches; red circles indicate unmatched points.

FAST and Rotated BRIEF). Both these techniques are attractive because of their good performance and low cost. In this paper, we address several limitations of these techniques vis-a-vis SIFT, most notably the lack of rotational invariance in BRIEF. Our main contributions are:

- The addition of a fast and accurate orientation component to FAST.
- The efficient computation of oriented BRIEF features.
- Analysis of variance and correlation of oriented BRIEF features.
- A learning method for de-correlating BRIEF features under rotational invariance, leading to better performance in nearest-neighbor applications.

To validate ORB, we perform experiments that test the properties of ORB relative to SIFT and SURF, for both raw matching ability, and performance in image-matching applications. We also illustrate the efficiency of ORB by implementing a patch-tracking application on a smart phone. An additional benefit of ORB is that it is free from the licensing restrictions of SIFT and SURF.

2. Related Work

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, November 2011, pp. 2564–2571

Reference

Histograms of Oriented Gradients for Human Detection

Navneet Dalal and Bill Triggs
INRIA Rhône-Alps, 655 avenue de l'Europe, Montbonnot 38334, France
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, http://lear.inrialpes.fr

Abstract

We study the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient based descriptors, we show experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. We study the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so we introduce a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds.

1 Introduction

Detecting humans in images is a challenging task owing to their variable appearance and the wide range of poses that they can adopt. The first need is a robust feature set that allows the human form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination. We study the issue of feature sets for human detection, showing that locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets including wavelets [17,22]. The proposed descriptors are reminiscent of edge orientation histograms [4,5], SIFT descriptors [12] and shape contexts [1], but they are computed on a dense grid of uniformly spaced cells and they use overlapping local contrast normalizations for improved performance. We make a detailed study of the effects of various implementation choices on detector performance, taking "pedestrian detection" (the detection of mostly visible people in more or less upright poses) as a test case. For simplicity and speed, we use linear SVM as a baseline classifier throughout the study. The new detectors give essentially perfect results on the MIT pedestrian test set [18,17], so we have created a more challenging set containing over 1800 pedestrian images with a large range of poses and backgrounds.

We briefly discuss previous work on human detection in §2, give an overview of our method §3, describe our data sets in §4 and give a detailed description and experimental evaluation of each stage of the process in §5–6. The main conclusions are summarized in §7.

2 Previous Work

There is an extensive literature on object detection, but here we mention just a few relevant papers on human detection [18,17,22,16,20]. See [6] for a survey. Papageorgiou *et al* [18] describe a pedestrian detector based on a polynomial SVM using rectified Haar wavelets as input descriptors, with a parts (subwindow) based variant in [17]. Depoortere *et al* give an optimized version of this [2]. Gavrila & Philomen [8] take a more direct approach, extracting edge images and matching them to a set of learned exemplars using chamfer distance. This has been used in a practical real-time pedestrian detection system [7]. Viola *et al* [22] build an efficient moving person detector, using AdaBoost to train a chain of progressively more complex region rejection rules based on Haar-like wavelets and space-time differences. Ronfard *et al* [19] build an articulated body detector by incorporating SVM based limb classifiers over 1st and 2nd order Gaussian filters in a dynamic programming framework similar to those of Felzenszwalb & Huttenlocher [3] and Ioffe & Forsyth [9]. Mikolajczyk *et al* [16] use combinations of orientation-position histograms with binary-thresholded gradient magnitudes to build a parts based method containing detectors for faces, heads, and front and side profiles of upper and lower body parts. In contrast, our detector uses a simple architecture with a single detection window, but appears to give significantly higher performance on pedestrian images.

3 Overview of the Method

This section gives an overview of our feature extraction chain, which is summarized in fig. 1. Implementation details are postponed until §6. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. Similar features have seen increasing use over the past decade [4,5,12,15]. The basic idea is that

Orientation Histograms for Hand Gesture Recognition

William T. Freeman and Michal Roth
Mitsubishi Electric Research Labs
201 Broadway
Cambridge, MA 02139 USA
e-mail: {freeman, roth}@merl.com

From: IEEE Intl. Wkshp. on Automatic Face and Gesture Recognition, Zurich, June, 1995.

Abstract

We present a method to recognize hand gestures, based on a pattern recognition technique developed by McConnell [16] employing histograms of local orientation. We use the orientation histogram as a feature vector for gesture classfication and interpolation.

This method is simple and fast to compute, and offers some robustness to scene illumination changes. We have implemented a real-time version, which can distinguish a small vocabulary of about 10 different hand gestures. All the computation occurs on a workstation; special hardware is used only to digitize the image. A user can operate a computer graphic crane under hand gesture control, or play a game. We discuss limitations of this method.

For moving or "dynamic gestures", the histogram of the spatio-temporal gradients of image intensity form the analogous feature vector and may be useful for dynamic gesture recognition.

1 Introduction

Computer recognition of hand gestures may provide a more natural human-computer interface, allowing people to point, or rotate a CAD model by rotating their hands. Interactive computer games would be enhanced if the computer could understand players' hand gestures. Gesture recognition may even be useful to control household appliances.

We distinguish two categories of gestures: static and dynamic. A static gesture is a particular hand configuration and pose, represented by a single image. A dynamic gesture is a moving gesture, represented by a sequence of images. We focus on the recognition of static gestures, although our method generalizes in a natural way to dynamic gestures.

For the broadest possible application, a gesture recognition algorithm should be fast to compute. Here, we apply a simple pattern recognition method to hand gesture recognition, resulting in a fast, useable hand gesture recognition system.

1.1 Related Work

The trackball, the joystick, and the mouse are extremely successful devices for hand-based computer input. Yet all require that the user hold some hard-

the special glove. We seek a visually based method which will be free of gloves and wires.

Relying on visual markings on the hands, previous researchers have recognized sign language and pointing gestures [24, 5, 8]. However, these methods require the placement of markers on the hands.

The marking-free systems of [12, 21] can recognize specific finger or pointing events, but not general gestures. Employing special hardware or off–line learning, several researchers have developed successful systems to recognize general hand gestures [22, 14, 6, 7, 20]. Blake and Isard [4] have developed a fast contour-based tracker which they applied to hands, but the discrimination of different hand poses is limited. The real-time hand gesture recognition systems we are aware of require special hardware or lengthy training analysis.

2 Our Approach

We seek a simple and fast algorithm, which works in real-time on a workstation. We want the recognition to be relatively robust to changes in lighting.

A high level approach might employ models of the hand, fingers, joints, and perhaps fit such a model to the visual data. This approach offers power and robustness, but at the expense of speed.

A low-level approach, such as was taken by [6], would process data at a level not much higher than that of pixel intensities. This approach would not have the power to make inferences about occluded data. However, it could be simple and fast. We chose this approach.

We use a pattern recognition system of the form of Fig. (1). Some transformation, T, converts an image or image sequence into a feature vector, which we then compare with the feature vectors of a training set of gestures. We will use a Euclidean distance metric.

We seek the simplest possible transformation T which allows recognition of hand gestures. To motivate the algorithm, let us first examine a transformation which is too simple, and then fix it.

Suppose we did nothing as our transformation, T and used the image itself as our feature vector. We would sum the squares of image pixel differences to measure distances between gestures. Figure 2 illustrates a problem with this scheme, and suggests a solution. (a) and (b) show the same hand gesture un-