İTÜ

3D Vision
BLG 634E

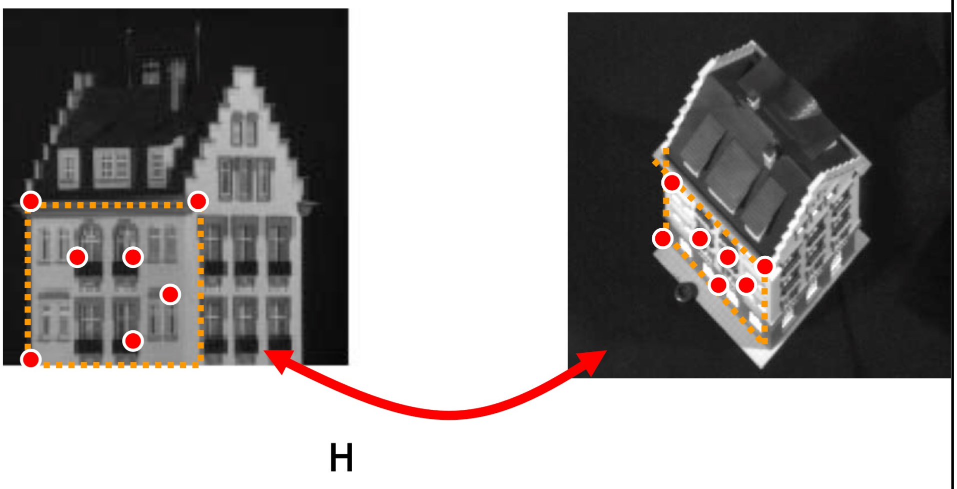
Professor: Gozde UNAL

Feature Extraction

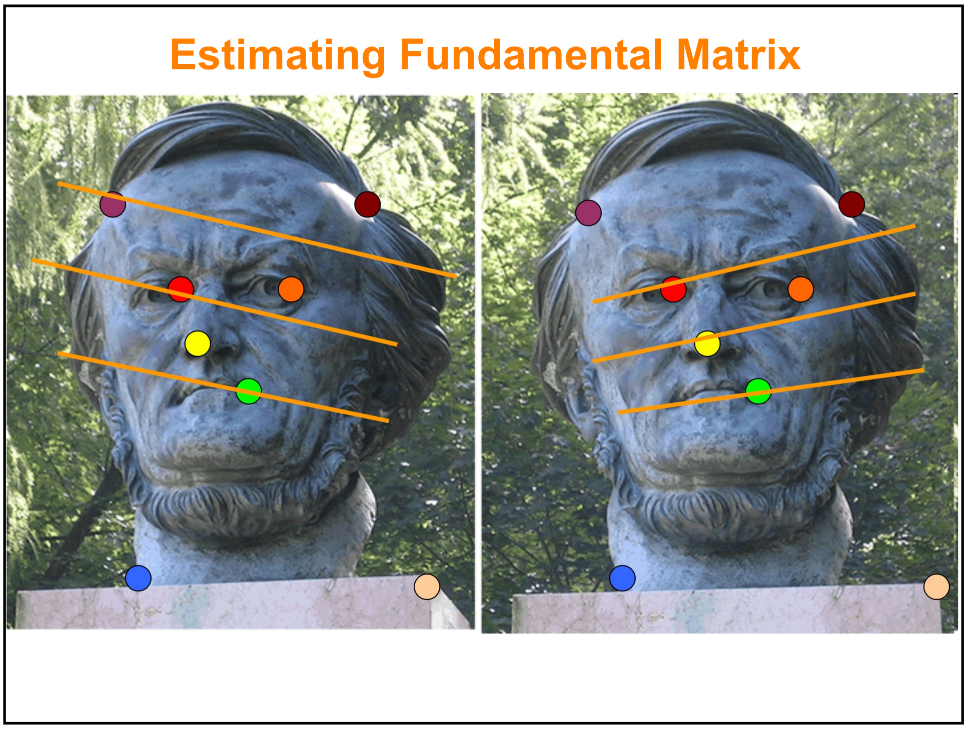
Some slides are from **Profs. Greg Slabaugh and S. Jalali** @ City University London

1

Recall: Estimating a Homography

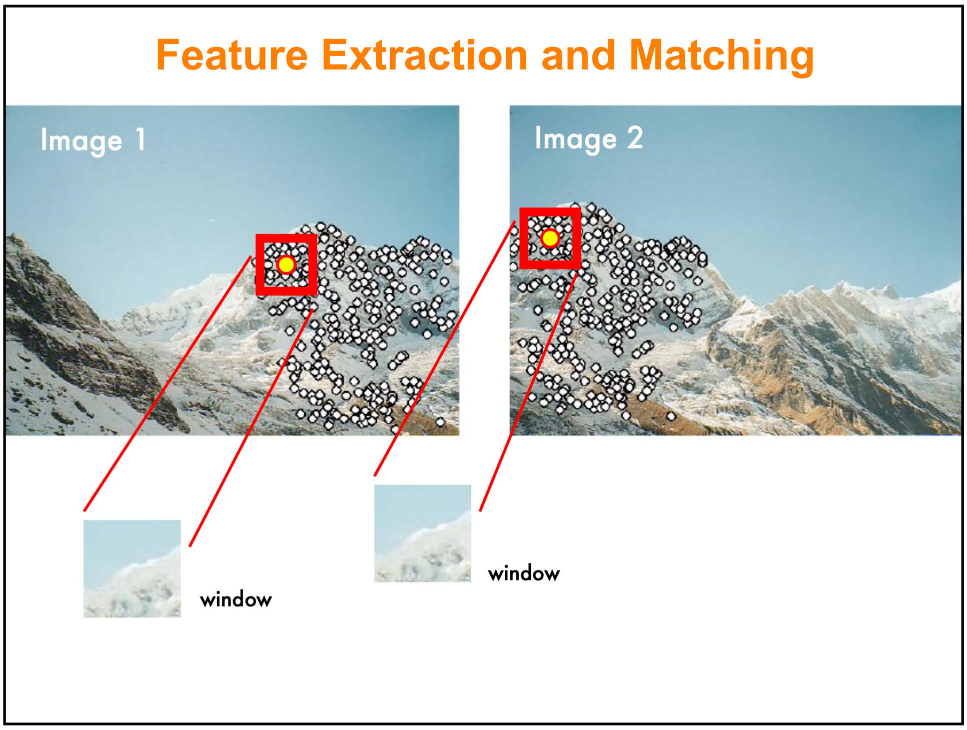


4



Estimating Fundamental Matrix

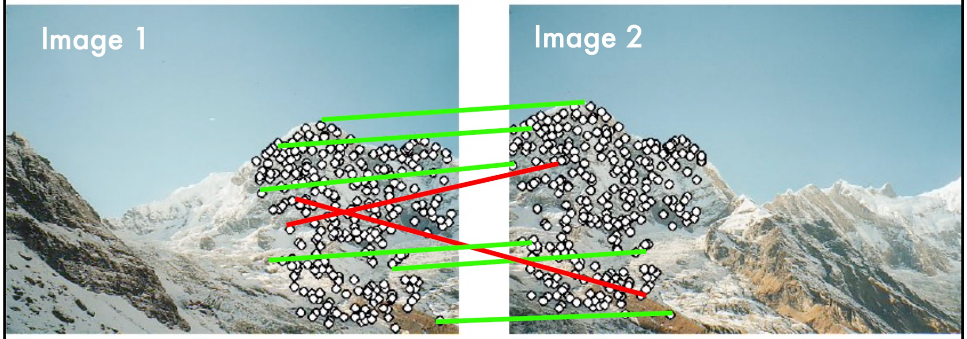
5



Feature Extraction and Matching

6

Feature Extraction and Matching

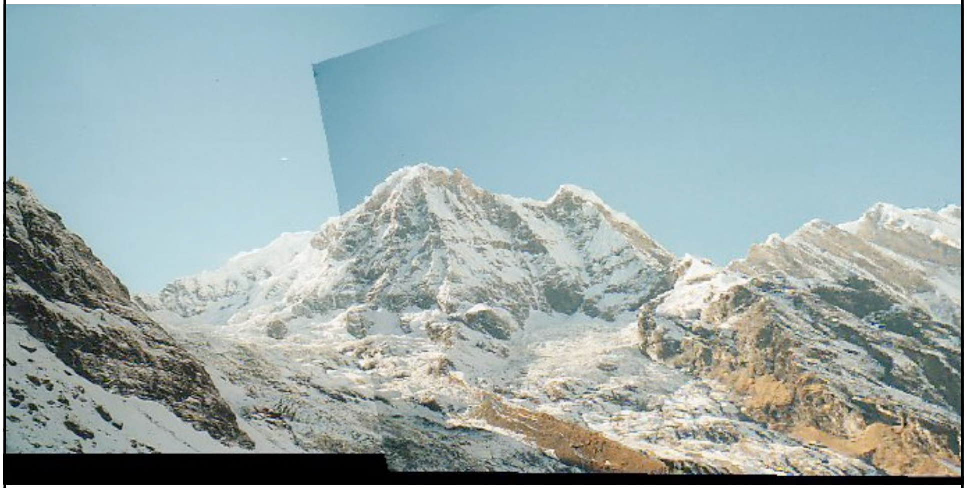


Matches based on appearance only

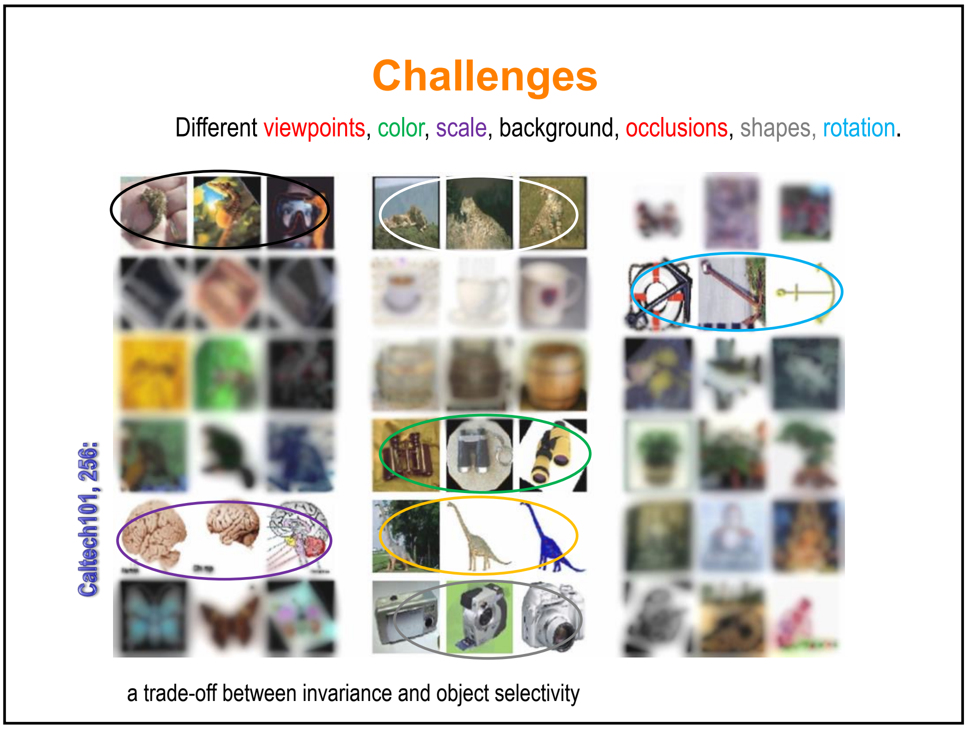
Green: good matches
Red: bad matches

7

Estimating a homography for Panorama



8



Challenges

Different viewpoints, color, scale, background, occlusions, shapes, rotation.

a trade-off between invariance and object selectivity

9

Solution

What features can be used: Color features? Shape features?

- Scale Invariant? Shift Invariant?

a) Caltech101 b) Scenes c)Soccer d) Flowers

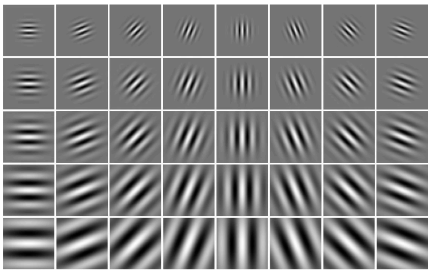
10

Feature Extraction

- Image Features (some examples)
  - Corner Features: Harris, Min eigenvalue etc.
  - HOG (Histogram of Oriented Gradients)
  - Local Binary Patterns
  - Bag of features
  - SIFT (Scale Invariant Feature Transform)
  - SURF (Speeded Up Robust Features)
  - Others: Blob, Gabor, Haar like features, …
  - MSER: maximal stable extremal region (region growing idea)

11

Gabor Filters



- Frequency and orientation representations of Gabor filters are inspired by those of the human visual system. Used for mainly texture representation and discrimination
- In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a complex sinusoid. The real component is modulated by a sinusoidal plane wave as here:

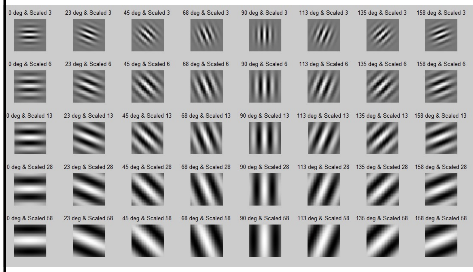
$$G(x,y) = \exp\left(-\frac{(X^2+\gamma^2 Y^2)}{2\sigma^2}\right)\cos\left(\frac{2\pi}{\lambda}X\right)$$

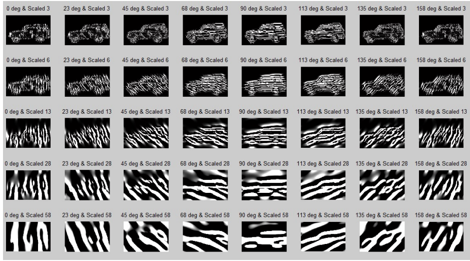
Teta: amount of rotation

$$X = x\cos\theta + y\sin\theta;\ \ Y = -x\sin\theta + y\cos\theta$$

12

Gabor Filters



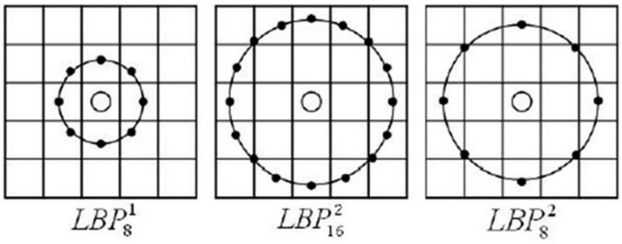


http://www.mathworks.com/matlabcentral/fileexchange/44630-gabor-feature-extraction

13

Local Binary Patterns Concept

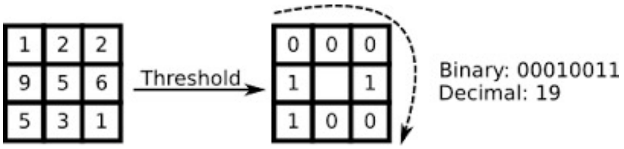
- Divide the examined window to cells (e.g. 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.



14

Local Binary Patterns Concept

- Where the center pixel's value is greater than the neighbor, write "1". Otherwise, write "0". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
- Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center).
- Optionally normalize the histogram.
- Concatenate normalized histograms of all cells. This gives the feature vector for the window.



1	2	2
9	5	6
5	3	1

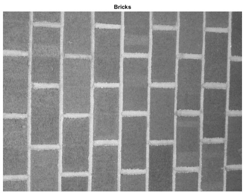
Threshold

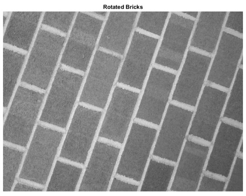
0	0	0
1		1
1	0	0

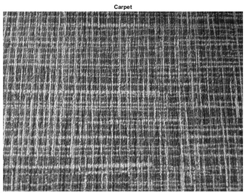
Binary: 00010011
Decimal: 19

15

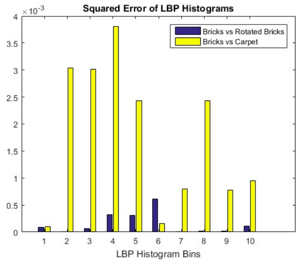
Matlab Example







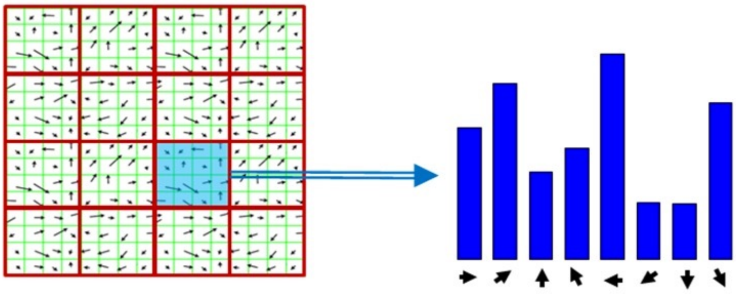
```
brickWall = imread('bricks.jpg');
rotatedBrickWall = imread('bricksRotated.jpg');
carpet = imread('carpet.jpg');

lbpBricks1 = extractLBPFeatures(brickWall,'Upright',false)
lbpBricks2 =
extractLBPFeatures(rotatedBrickWall,'Upright',false);
lbpCarpet = extractLBPFeatures(carpet,'Upright',false);

brickVsBrick = (lbpBricks1 - lbpBricks2).^2;
brickVsCarpet = (lbpBricks1 - lbpCarpet).^2;
figure bar([brickVsBrick; brickVsCarpet]','grouped')
title('Squared Error of LBP Histograms') xlabel('LBP
Histogram Bins')
legend('Bricks vs Rotated Bricks','Bricks vs Carpet’)
```



16

Histogram of Oriented Gradients

- The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection.
- The technique counts occurrences of gradient orientation in localized portions of an image.
- Similar to that of scale-invariant feature transform descriptors (SIFT) and Speeded Up Robust Features (SURF) but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

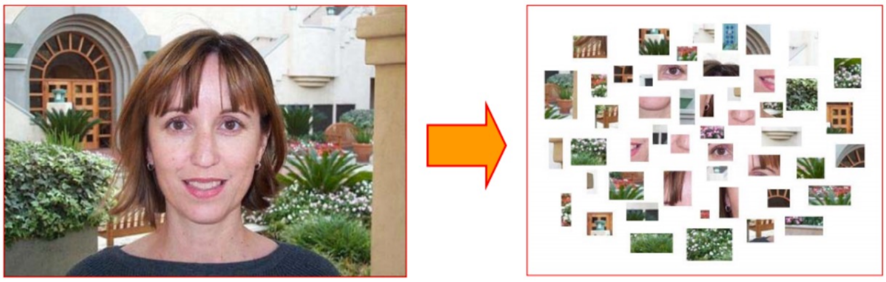


https://www.youtube.com/watch?v=0Zib1YEE4LU&feature=plcp

18

Dictionary of Features Models

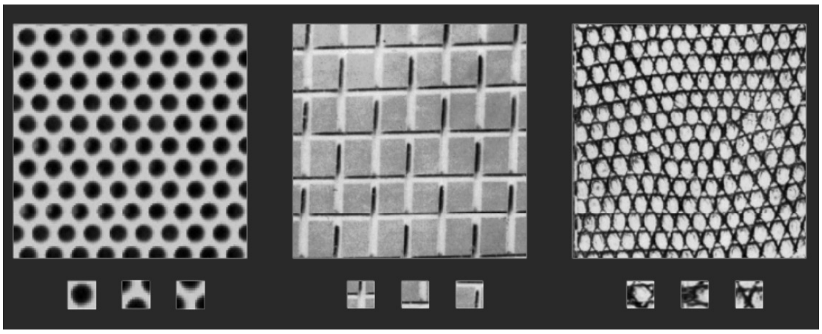
Motivated from Bag of Words: Classify text in terms of frequency of words in a dictionary



19

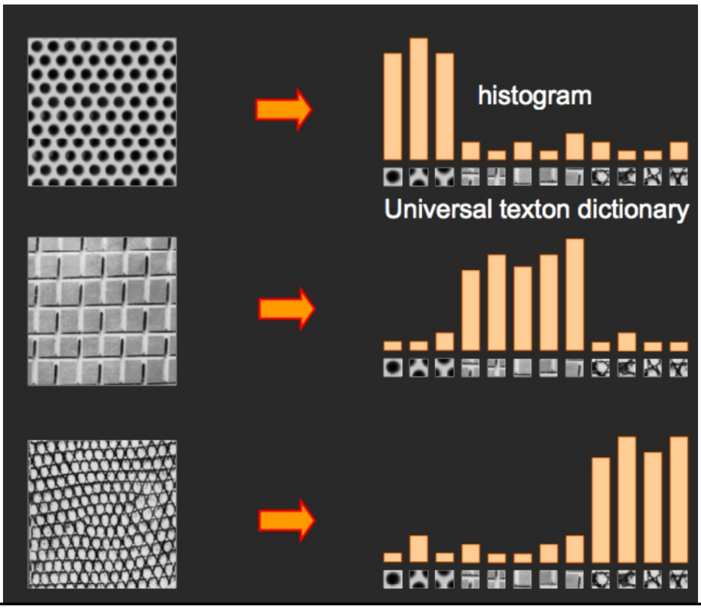
Bag of Features Models

- Texture is characterized by the repetition of basic elements or textons (dictionary elements)
- the identity of textons, not their spatial arrangement, matters



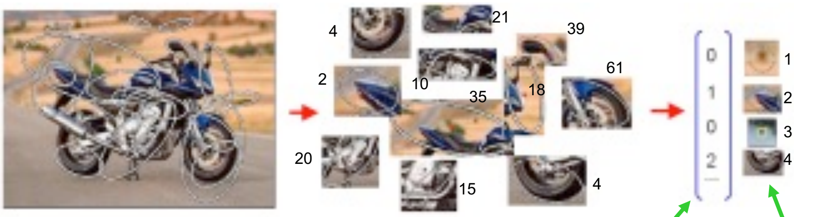
20

Bag of Features Models



21

Bag of features (words)



Interest regions

Visual words

Histogram

Dictionary

Stack visual word histograms
as columns in matrix

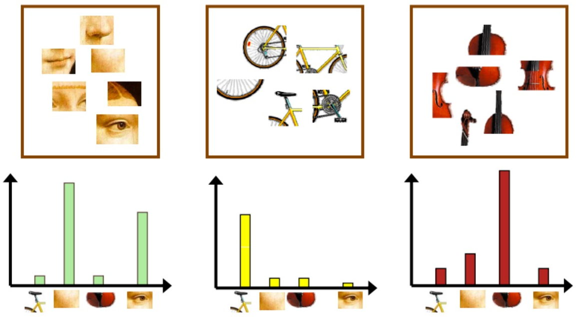
Throw away spatial information!

https://www.youtube.com/watch?v=iGZpJZhqEME&feature=plcp

22

Bag of Features

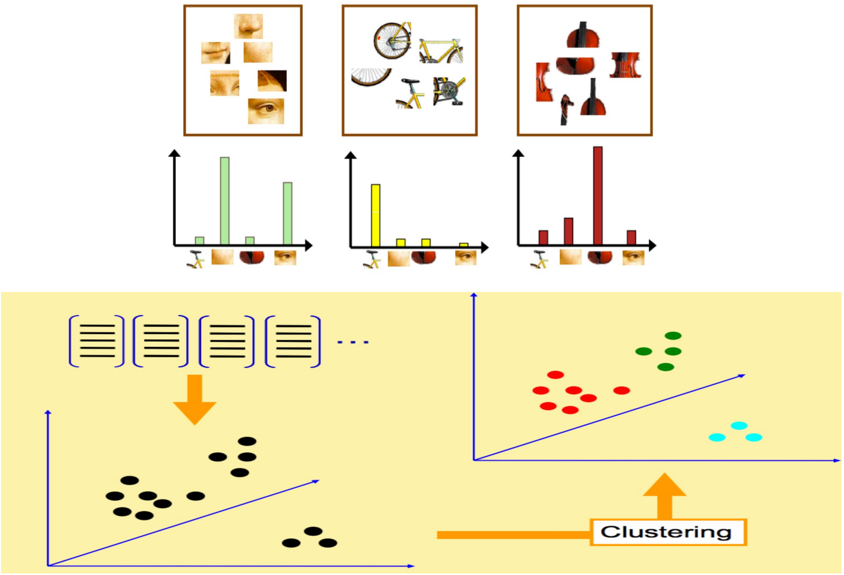
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”



Note that for the bag of features approach to be effective, majority of each image's area must be occupied by the subject of the category, for example, an object or a type of scene.

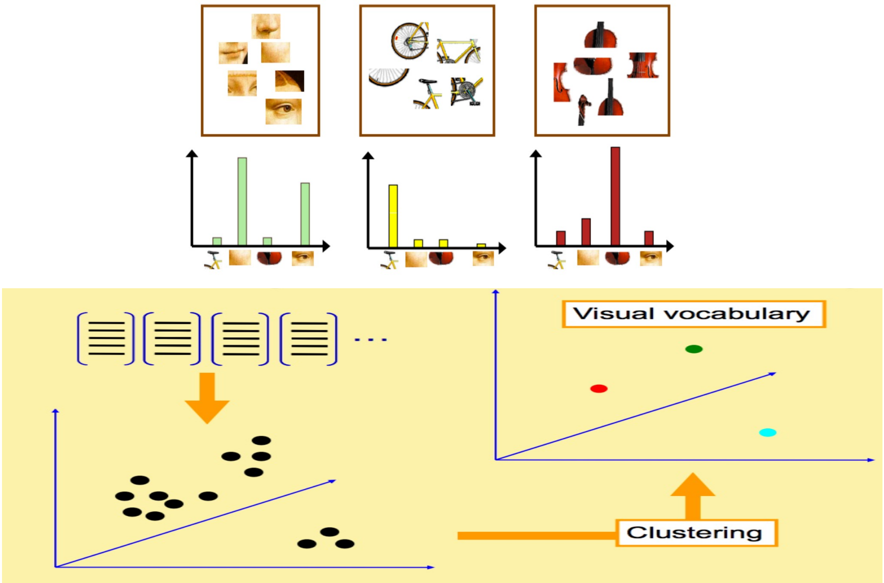
23

Learning the Visual Vocabulary

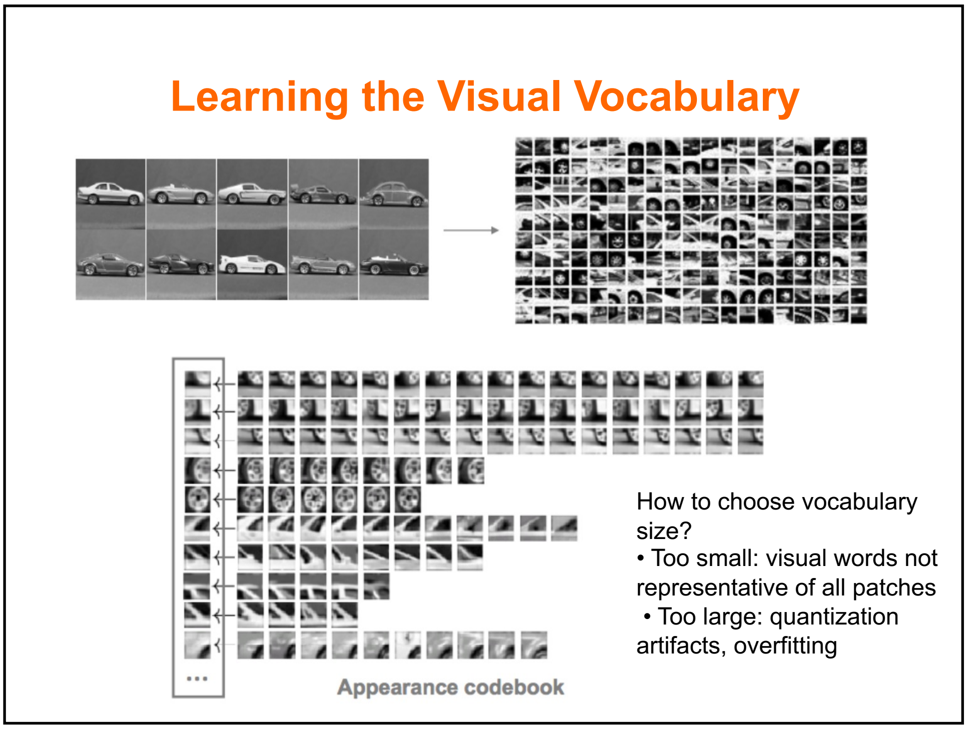


24

Learning the Visual Vocabulary



25

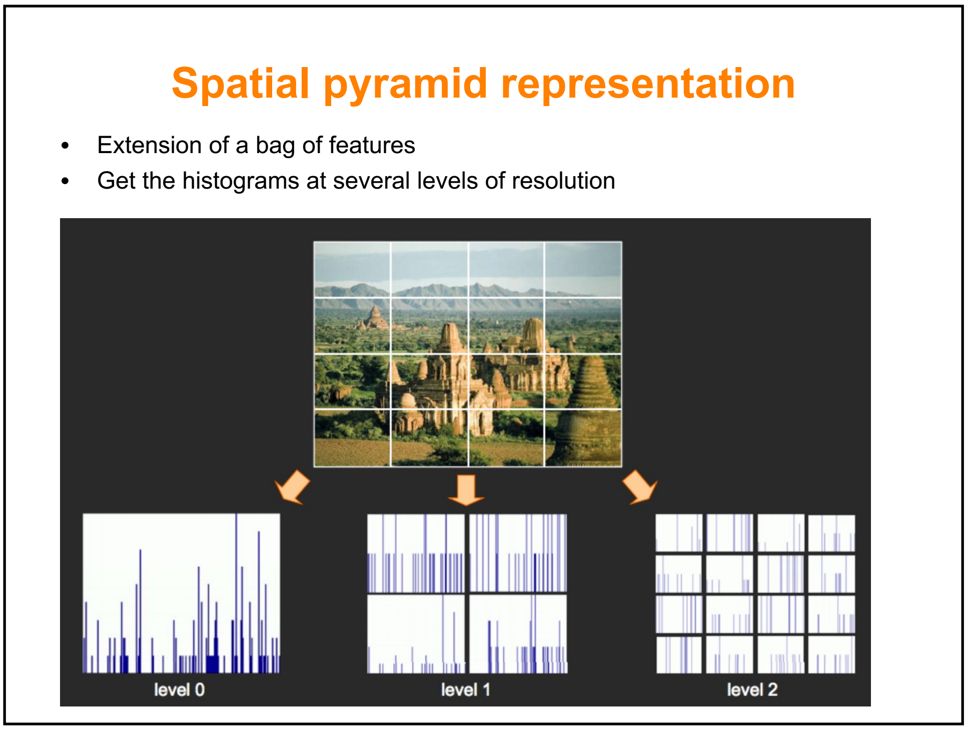


Learning the Visual Vocabulary

How to choose vocabulary size?

- Too small: visual words not representative of all patches
- Too large: quantization artifacts, overfitting

26



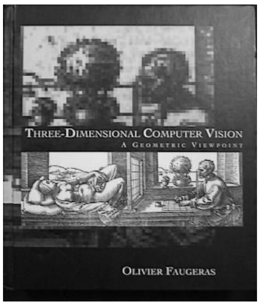
Spatial pyramid representation

- Extension of a bag of features
- Get the histograms at several levels of resolution

27

Let’s Return to this Problem…

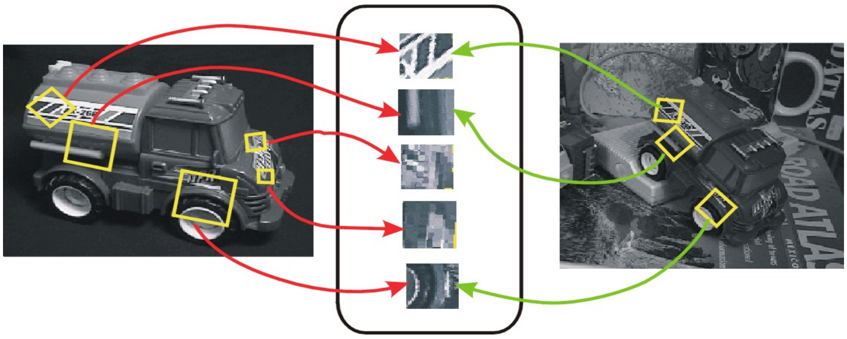
Want to find



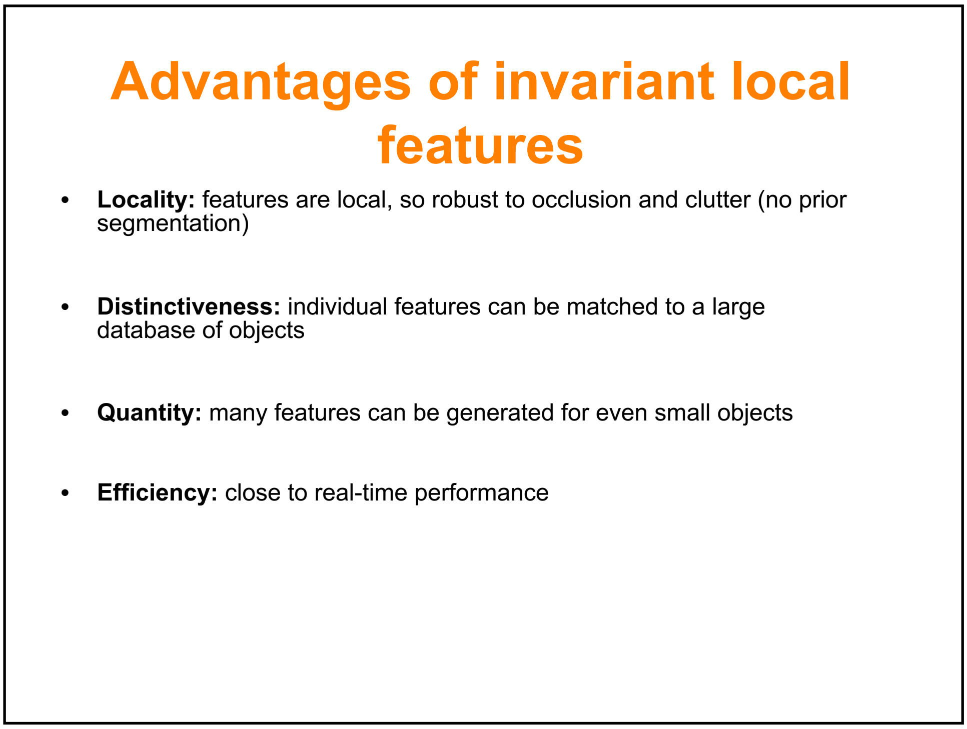
… in here

Invariant Local Features

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



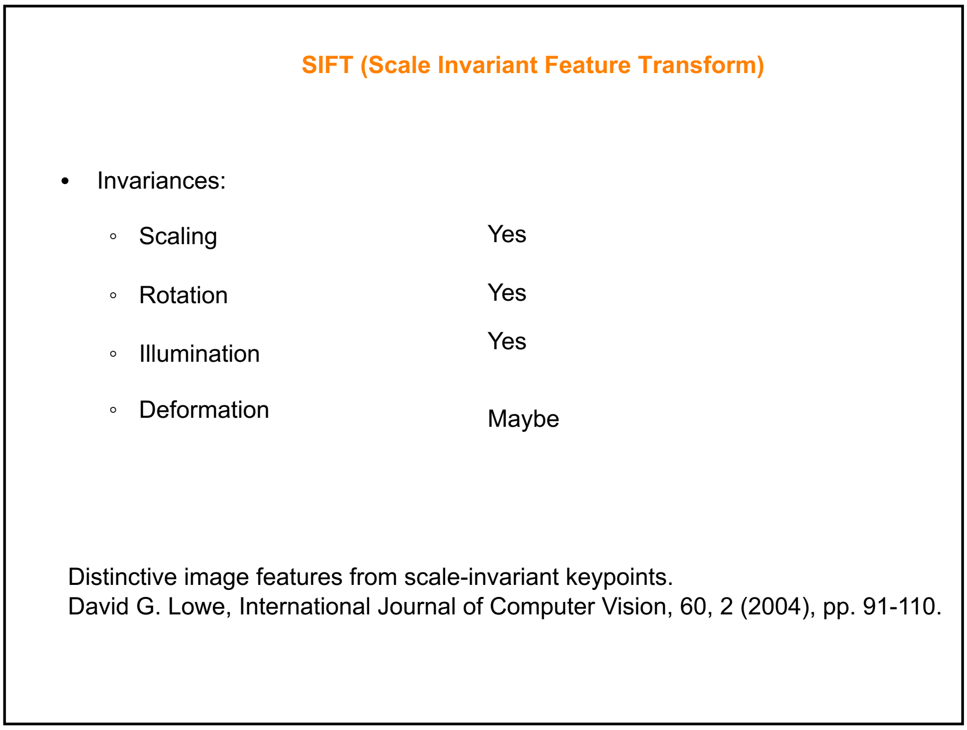
SIFT Features



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

30



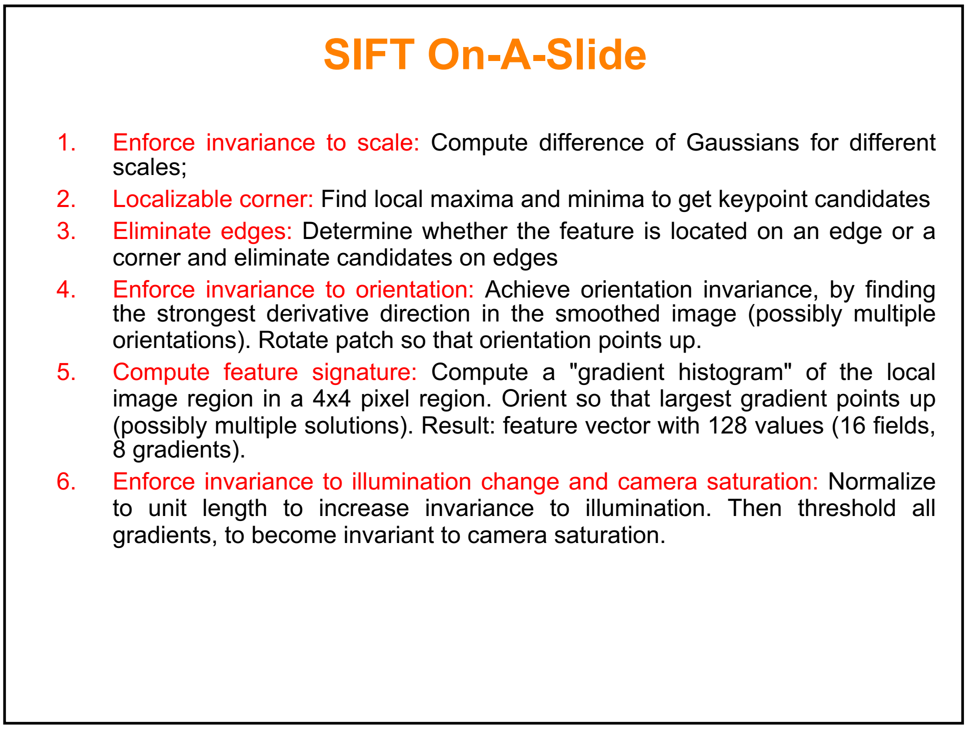
SIFT (Scale Invariant Feature Transform)

- Invariances:

◦ Scaling	Yes
◦ Rotation	Yes
◦ Illumination	Yes
◦ Deformation	Maybe

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

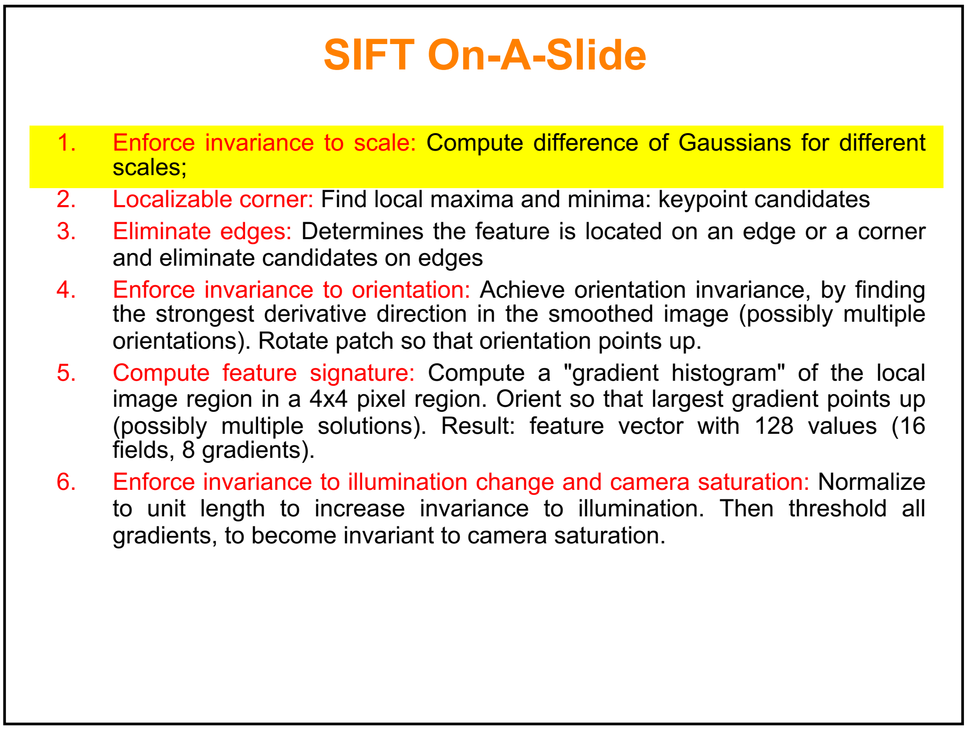
31



SIFT On-A-Slide

1. Enforce invariance to scale: Compute difference of Gaussians for different scales;
2. Localizable corner: Find local maxima and minima to get keypoint candidates
3. Eliminate edges: Determine whether the feature is located on an edge or a corner and eliminate candidates on edges
4. Enforce invariance to orientation: Achieve orientation invariance, by finding the strongest derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (16 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

32



SIFT On-A-Slide

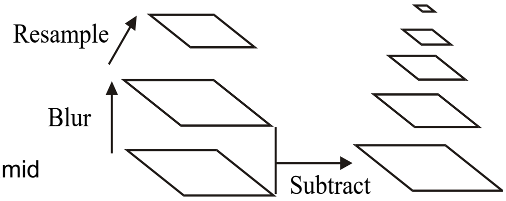
1. Enforce invariance to scale: Compute difference of Gaussians for different scales;
2. Localizable corner: Find local maxima and minima: keypoint candidates
3. Eliminate edges: Determines the feature is located on an edge or a corner and eliminate candidates on edges
4. Enforce invariance to orientation: Achieve orientation invariance, by finding the strongest derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (16 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

33

Build Scale-Space Pyramid

- Scale space is separated into octaves:
  - Octave 1 uses scale σ
  - Octave 2 uses scale 2σ, etc.
  - In each octave, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images.
  - Adjacent Gaussians are subtracted to produce the DoG (difference of Gaussians is a feature enhancement algorithm that involves the subtraction of one blurred version of an original image from another, less blurred version of the original)
  - After each octave, the Gaussian image is down-sampled by a factor of 2 to produce an image ¼ the size to start the next level.

Idea:
Find Corners, but scale invariance
Approach:
Run linear filter (diff of Gaussians)
At different resolutions of image pyramid



34

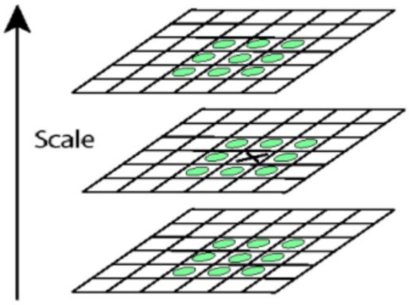
SIFT On-A-Slide

1. Enforce invariance to scale: Compute difference of Gaussians for different scales;
2. Localizable corner: Find local maxima and minima: keypoint candidates
3. Eliminate edges: Determines the feature is located on an edge or a corner and eliminate candidates on edges
4. Enforce invariance to orientation: Achieve orientation invariance, by finding the strongest derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (16 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

35

Keypoint Localization

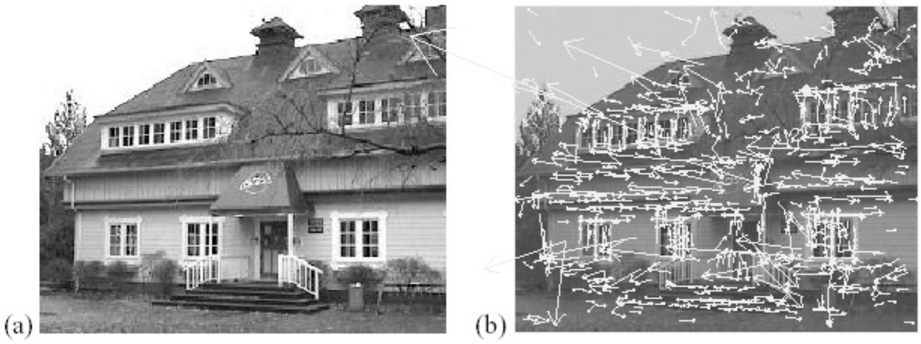
- Detect maxima and minima of difference of Gaussian in scale space □
- Each point is compared to its 8 neighbours in the current image and 9 neighbours each in the scales above and below



For each max or min found, output is the **location** and the **scale**.

36

Example of keypoint detection



(a) 233x189 image
(b) 832 DOG extrema

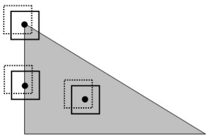
37

Example of keypoint detection

To eliminate undesired keypoints (i.e. edges), SIFT algorithm uses 2x2 Hessian matrix.

- It determines the feature is located on an edge or a corner. For a stable feature the curvature across a feature point should be high in more than one direction.

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



- A feature with change in only one direction is unstable. A large difference between principal curvatures across a feature is therefore bad.

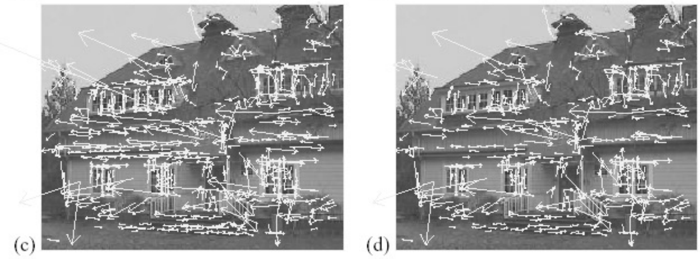
- The Hessian matrix gives a short cut so that the principal curvatures (eigenvalues of the shape operator at the point) across the keypoint do not have to be explicitly calculated to determine if the change in gradient is large in more than one direction.

- Instead the ratio between the Eigen values of the Hessian matrix can be used. This reduces processing and increases speed.

The Hessian matrix or Hessian is a square matrix of second-order partial derivatives of a scalar-valued function, or scalar field. It describes the local curvature of a function of many variables.

Example of keypoint detection

Threshold on value at DoG peak and on ratio of principle curvatures (Similar to Harris corner detector approach)



(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures

SIFT On-A-Slide

1. Enforce invariance to scale: Compute difference of Gaussians for different scales;
2. Localizable corner: Find local maxima and minima: keypoint candidates
3. Eliminate edges: Determines the feature is located on an edge or a corner and eliminate candidates on edges
4. Enforce invariance to orientation: Achieve orientation invariance, by finding the strongest derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (16 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

Select canonical orientation

- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)

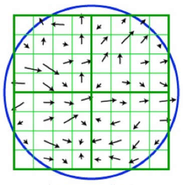
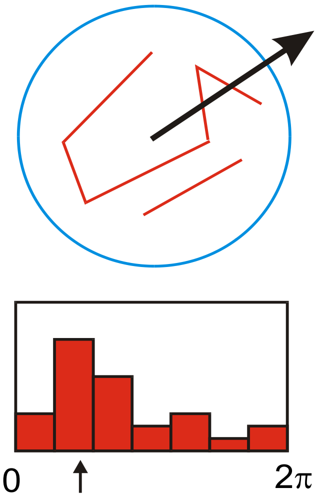


Image gradients



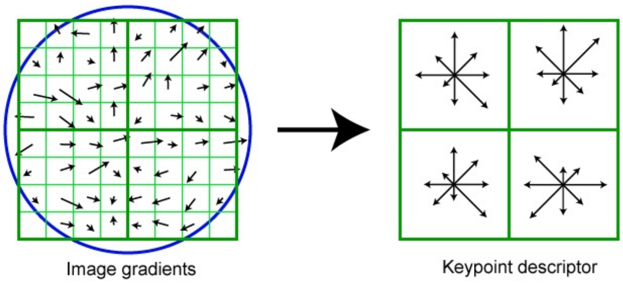
SIFT On-A-Slide

1. Enforce invariance to scale: Compute difference of Gaussians for different scales;
2. Localizable corner: Find local maxima and minima: keypoint candidates
3. Eliminate edges: Determines the feature is located on an edge or a corner and eliminate candidates on edges
4. Enforce invariance to orientation: Achieve orientation invariance, by finding the strongest derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (16 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

42

SIFT vector formation

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



SIFT Matlab Toolbox: http://www.vlfeat.org/overview/sift.html

43

SIFT descriptor vector

- 4x4 Gradient window
- Histogram of 4x4 samples per window in 8 directions
- Gaussian weighting around center(σ is 0.5 times that of the scale of a keypoint)
- 4x4x8 = 128 dimensional feature vector

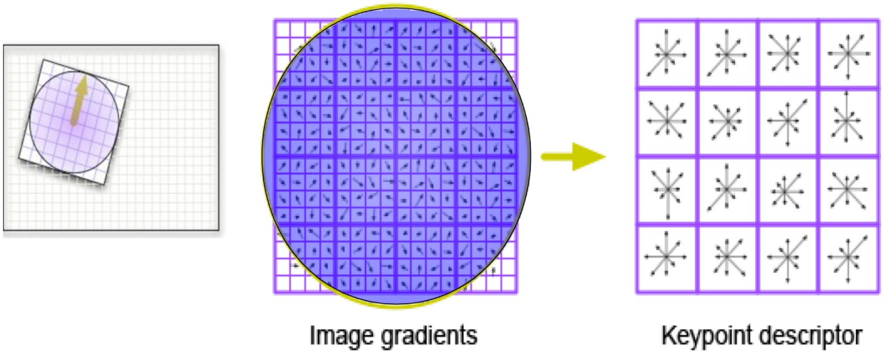
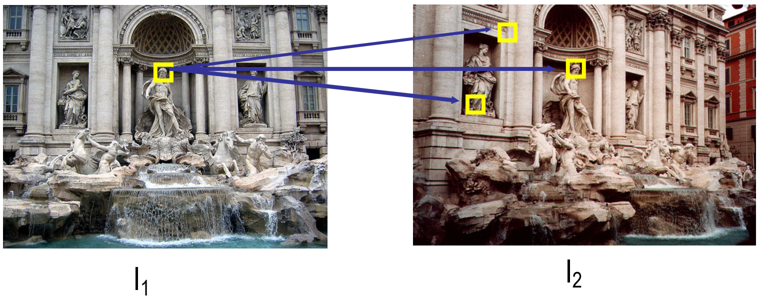


Image from: Jonas Hurrelmann

Matching SIFT features

- **Given a feature in I_1, how to find the best match in I_2?**
  1. Define distance function that compares two descriptors.
  2. Test all the features in I_2, find the one with min distance.
     a good distance measure is sum of squared difference (SSD) which is equivalent to the squared L2-norm, also known as Euclidean norm.) $\sum(x_i-y_i)^2$



More on Distance Metrics: http://numerics.mathdotnet.com/Distance.html

SURF

- Speeded Up Robust Features (SURF) is a local feature detector and descriptor
- Inspired by SIFT.
- The standard version of SURF is several times (x3: you check?) faster than SIFT.
- Claimed by its authors to be more robust against different image transformations than SIFT.
- SURF is not as well as SIFT on invariance to illumination change and viewpoint change

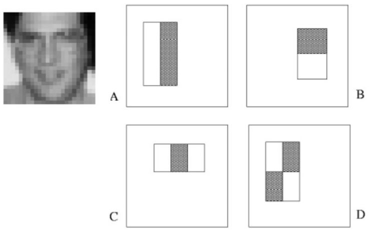
- Speed-up computations by fast approximation of
  (i) Hessian matrix and
  (ii) descriptor using “integral images”.

See the SURF paper for details:
Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, “SURF: Speeded Up Robust Features”, **European Computer Vision Conference (ECCV),** 2006.

46

Feature Extraction

“Rectangle filters”



Value =

∑ (pixels in white area) –
∑ (pixels in black area)

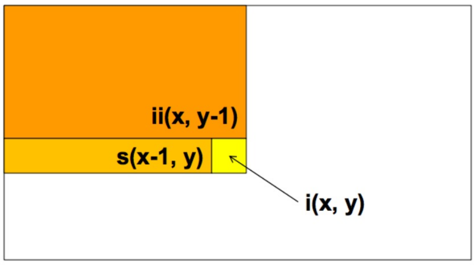
So many computations…

→ Integral images for fast feature evaluation

47

Integral Images

- The integral image computes a value at each pixel (x,y) that is the sum of the pixel values above and to the left of (x y) inclusive
- This can quickly be computed in one pass through the image



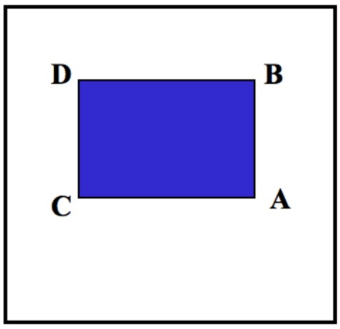
- Cumulative row sum:
- s(x, y) = s(x –1, y) + i(x, y)
- Integral image: ii(x, y) = ii(x, y −1) + s(x, y)

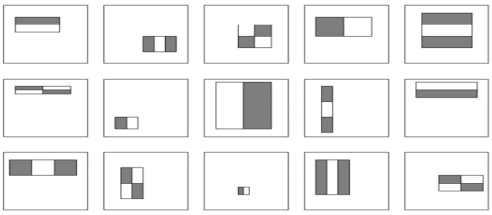
- MATLAB:
  - J = integralImage(I)

Integral Images

- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

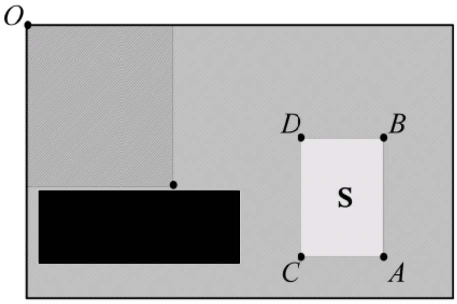
  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!





Integral Image

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



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

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

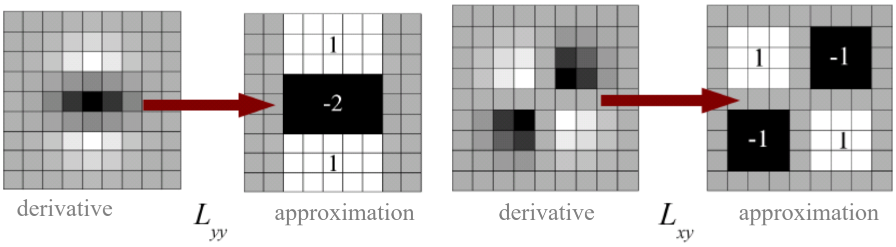
50

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

SURF: Speeded Up Robust Features

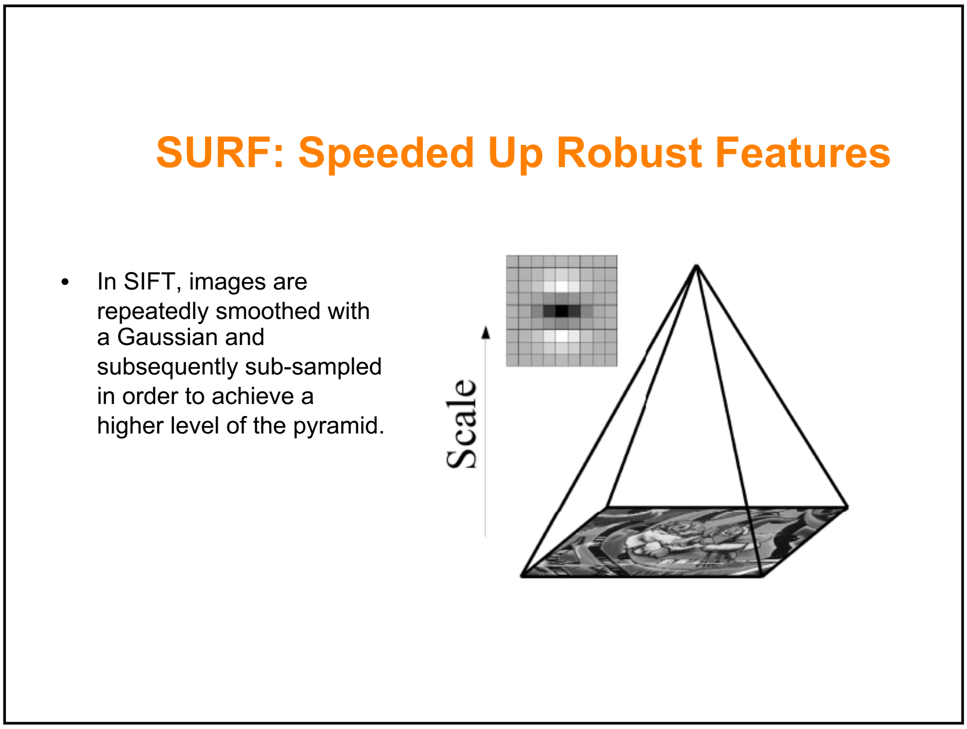
- Approximate L_{xx}, L_{yy}, and L_{xy} using box filters.

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



- Can be computed very fast using integral images!

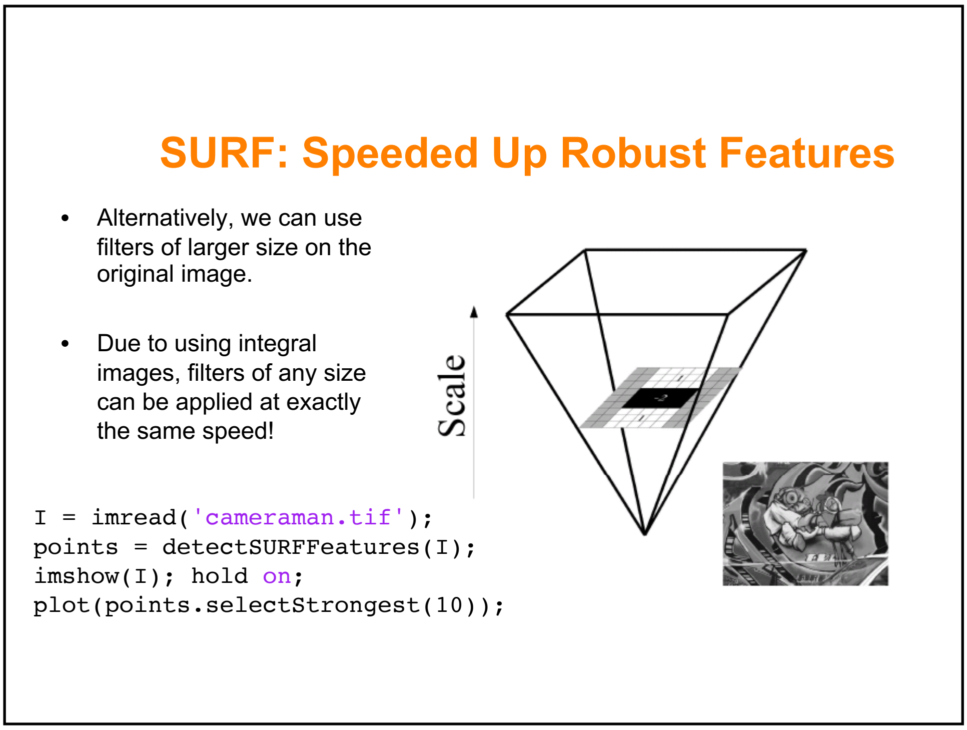
51



SURF: Speeded Up Robust Features

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

52



SURF: Speeded Up Robust Features

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

```
I = imread('cameraman.tif');
points = detectSURFFeatures(I);
imshow(I); hold on;
plot(points.selectStrongest(10));
```

53

Example application: Panoramic Images

SIFT/SURF feature matching can be used in image stitching for fully automated panorama reconstruction from non-panoramic images.

- We need to match (align) images
- Global methods sensitive to occlusion, lighting, parallax (camera displacement) effects. So look for local features that match well.

Panoramic Images

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images (e.g. Estimate geometric transformation such as homography)