Computer Vision – Lecture 8

Local Features

14.05.2019

Bastian Leibe
Visual Computing Institute
RWTH Aachen University
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

Course Outline

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition & Categorization
  - Sliding Window based Object Detection
- Local Features & Matching
  - Local Features – Detection and Description
  - Recognition with Local Features
- Deep Learning
- 3D Reconstruction

Recap: Sliding-Window Object Detection

- If object may be in a cluttered scene, slide a window around looking for it.

→ Car/non-car Classifier

- Essentially, this is a brute-force approach with many local decisions.

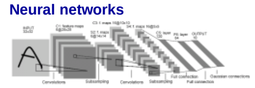
Slide credit: Kristen Grauman

Classifier Construction: Many Choices…

Nearest Neighbor

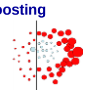
Berg, Berg, Malik 2005, Chum, Zisserman 2007, Boiman, Shechtman, Irani 2008, …

Neural networks



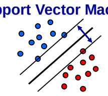
LeCun, Bottou, Bengio, Haffner 1998 Rowley, Baluja, Kanade 1998 …

Boosting



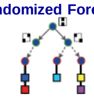
Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006, Benenson 2012, …

Support Vector Machines



Vapnik, Schölkopf 1995, Papageorgiou, Poggio '01, Dalal, Triggs 2005, Vedaldi, Zisserman 2012

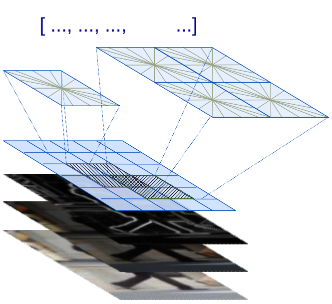
Randomized Forests



Amit, Geman 1997, Breiman 2001, Lepetit, Fua 2006, Gall, Lempitsky 2009,…

Slide adapted from Kristen Grauman

Recap: HOG Descriptor Processing Chain



Object/Non-object
↑
Linear SVM
↑
Collect HOGs over detection window
↑
Contrast normalize over overlapping spatial cells
↑
Weighted vote in spatial & orientation cells
↑
Compute gradients
↑
Gamma compression
↑
Image Window

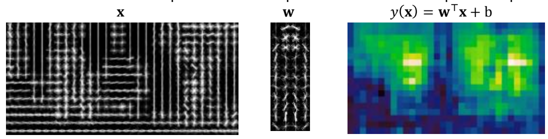
Slide credit: Navneet Dalal

Recap: Pedestrian Detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with learned template $\mathbf{w}$
  - Linear SVM classification function

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$$

HOG feature map $\mathbf{x}$ | Template $\mathbf{w}$ | Detector response map $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$

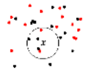


N. Dalal and B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Slide credit: Svetlana Lazebnik

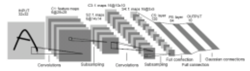
Classifier Construction: Many Choices…

Nearest Neighbor



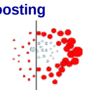
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005,
Boiman, Shechtman, Irani 2008, …

Neural networks



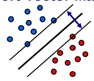
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Boosting



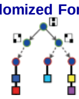
Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,
Benenson 2012, …

Support Vector Machines



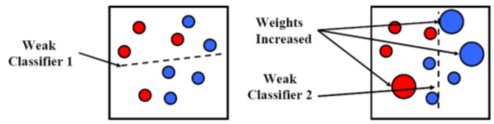
Vapnik, Schölkopf 1995,
Papageorgiou, Poggio '01,
Dalal, Triggs 2005,
Vedaldi, Zisserman 2012

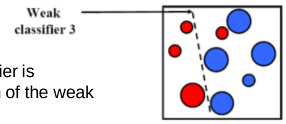
Randomized Forests



Amit, Geman 1997,
Breiman 2001,
Lepetit, Fua 2006,
Gall, Lempitsky 2009,…

Recap: AdaBoost

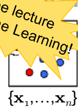




Final classifier is combination of the weak classifiers

AdaBoost: Detailed Training Algorithm

see lecture Machine Learning!



1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1,\ldots,N$.
2. For $m = 1,\ldots,M$ iterations
   a) Train a new weak classifier $h_m(\mathbf{x})$ using the current weighting coefficients $\mathbf{W}^{(m)}$ by minimizing the weighted error function
   $$J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n) \qquad I(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}$$
   b) Estimate the weighted error of this classifier on $\mathbf{X}$:
   $$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$
   c) Calculate a weighting coefficient for $h_m(\mathbf{x})$:
   $$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$
   d) Update the weighting coefficients:
   $$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(h_m(\mathbf{x}_n) \neq t_n)\}$$

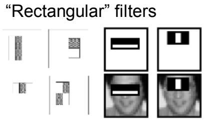
Example: Face Detection

- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a "patch"/window

- Now we'll take AdaBoost and see how the Viola-Jones face detector works

Feature extraction

"Rectangular" filters

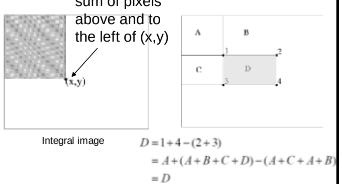


Feature output is difference between adjacent regions

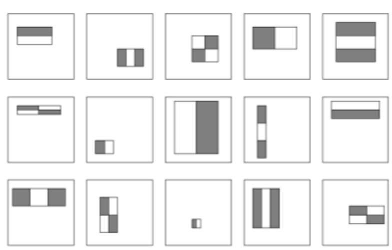
Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost

Value at (x,y) is sum of pixels above and to the left of (x,y)



Large Library of Features



Considering all possible filter parameters: position, scale, and type:

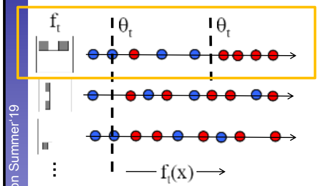
180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

Weak classifier: feature output > θ?

AdaBoost for Feature+Classifier Selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

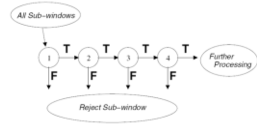
$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

For next round, reweight the examples according to errors, choose another filter/threshold combo.

[Viola & Jones, CVPR 2001]

Cascading Classifiers for Detection

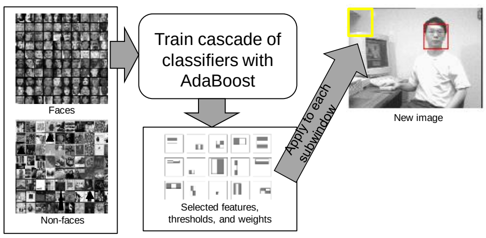
- Even if the filters are fast to compute, each new image has a lot of possible windows to search.
- For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,
  - Filter for promising regions with an initial inexpensive classifier
  - Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain



[Fleuret & Geman, IJCV 2001]
[Rowley et al., PAMI 1998]
[Viola & Jones, CVPR 2001]

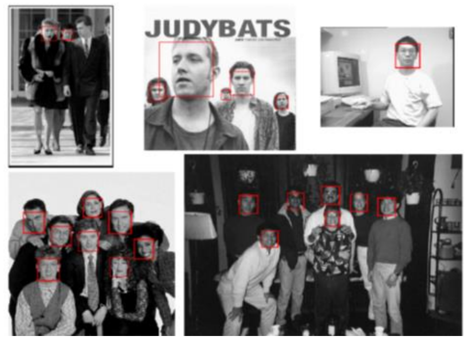
Figure from Viola & Jones CVPR

Recap: Viola-Jones Face Detector



- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]

Viola-Jones Face Detector: Results



Viola-Jones Face Detector: Results

You Can Try It At Home…

- The Viola & Jones detector was a huge success
  - First real-time face detector available
  - Many derivative works and improvements
- C++ implementation available in OpenCV [Lienhart, 2002]
  - http://sourceforge.net/projects/opencvlibrary/
- Matlab wrappers for OpenCV code available, e.g. here
  - http://www.mathworks.com/matlabcentral/fileexchange/19912

P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004

Limitations: Changing Aspect Ratios

- Sliding window requires fixed window size
  - Basis for learning efficient cascade classifier
- How to deal with changing aspect ratios?
  - Fixed window size

    ⇒ Wastes training dimensions

  - Adapted window size

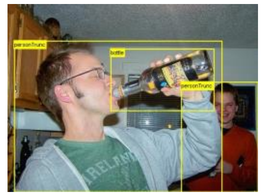
    ⇒ Difficult to share features

  - “Squashed” views [Dalal&Triggs]

    ⇒ Need to squash test image, too

Limitations (continued)

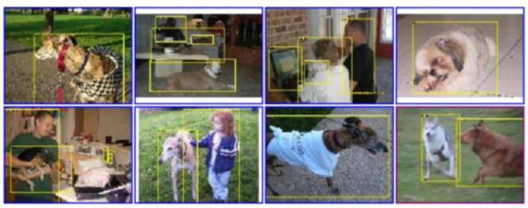
- Not all objects are “box” shaped



Slide credit: Kristen Grauman

Limitations (continued)

- Non-rigid, deformable objects not captured well with representations assuming a fixed 2D structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions



Slide credit: Kristen Grauman

Limitations (continued)

- If considering windows in isolation, context is lost

Sliding window

Detector’s view

Figure credit: Derek Hoiem

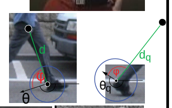
Topics of This Lecture

- Local Invariant Features
  - Motivation
  - Requirements, Invariances
- Keypoint Localization
  - Harris detector
  - Hessian detector
- Scale Invariant Region Selection
  - Automatic scale selection
  - Laplacian-of-Gaussian detector
  - Difference-of-Gaussian detector
  - Combinations
- Local Descriptors
  - Orientation normalization
  - SIFT

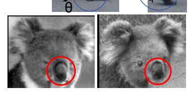
Motivation

- Global representations have major limitations
- Instead, describe and match only local regions
- Increased robustness to
  - Occlusions

  - Articulation

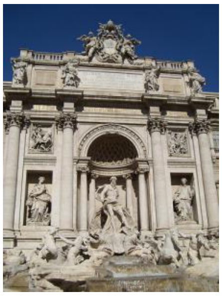


  - Intra-category variations



Application: Image Matching

by Diva Sian



by swashford

Slide credit: Steve Seitz

B. Leibe

28

Computer Vision Summer'19

Harder Case

by Diva Sian

by scgbt

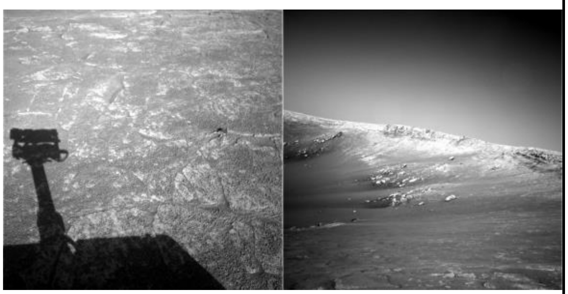
Slide credit: Steve Seitz

B. Leibe

29

Computer Vision Summer'19

Harder Still?



NASA Mars Rover images

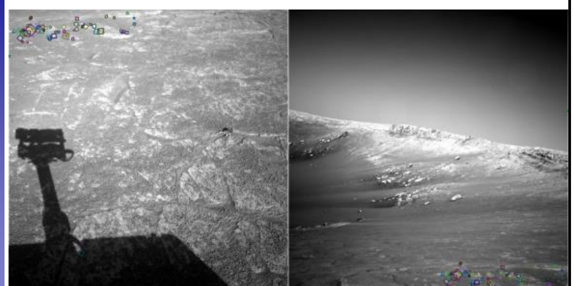
Slide credit: Noah Snavely

B. Leibe

30

Computer Vision Summer'19

Answer Below (Look for Tiny Colored Squares)



NASA Mars Rover images
with SIFT feature matches

Slide credit: Noah Snavely

B. Leibe

31

Computer Vision Summer'19

Application: Image Stitching

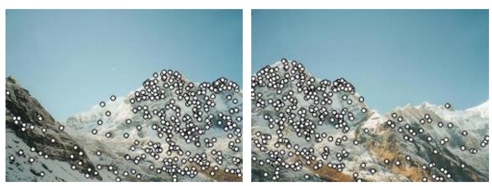
Slide credit: Darya Frolova, Denis Simakov

B. Leibe

32

Computer Vision Summer'19

Application: Image Stitching



- Procedure:
  - Detect feature points in both images

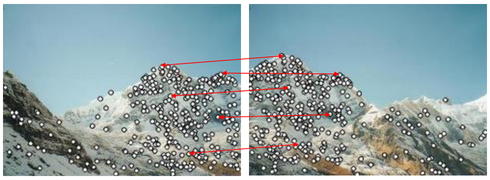
Slide credit: Darya Frolova, Denis Simakov

B. Leibe

33

Computer Vision Summer'19

Application: Image Stitching



- Procedure:
  - Detect feature points in both images
  - Find corresponding pairs

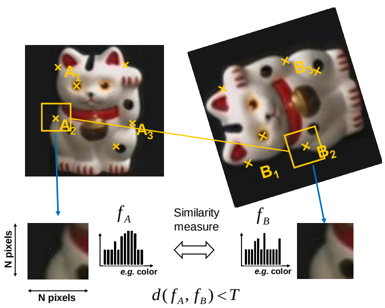
Slide credit: Darya Frolova, Denis Simakov

Application: Image Stitching

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

Slide credit: Darya Frolova, Denis Simakov

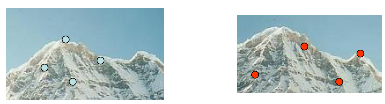
General Approach



1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

Common Requirements

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



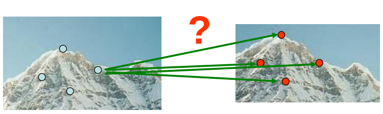
No chance to match!

We need a repeatable detector!

Slide credit: Darya Frolova, Denis Simakov

Common Requirements

- Problem 1:
  - Detect the same point *independently* in both images
- Problem 2:
  - For each point correctly recognize the corresponding one



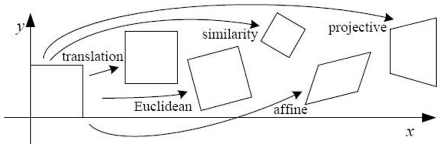
We need a reliable and distinctive descriptor!

Slide credit: Darya Frolova, Denis Simakov

Invariance: Geometric Transformations

Slide credit: Steve Seitz

Levels of Geometric Invariance



Requirements

- Region extraction needs to be repeatable and accurate
  - Invariant to translation, rotation, scale changes
  - Robust or covariant to out-of-plane (≈affine) transformations
  - Robust to lighting variations, noise, blur, quantization
- Locality: Features are local, therefore robust to occlusion and clutter.
- Quantity: We need a sufficient number of regions to cover the object.
- Distinctiveness: The regions should contain “interesting” structure.
- Efficiency: Close to real-time performance.

Many Existing Detectors Available

Hessian & Harris	[Beaudet ‘78], [Harris ‘88]
Laplacian, DoG	[Lindeberg ‘98], [Lowe ‘99]
Harris-/Hessian-Laplace	[Mikolajczyk & Schmid ‘01]
Harris-/Hessian-Affine	[Mikolajczyk & Schmid ‘04]
EBR and IBR	[Tuytelaars & Van Gool ‘04]
MSER	[Matas ‘02]
Salient Regions	[Kadir & Brady ‘01]
Others…	

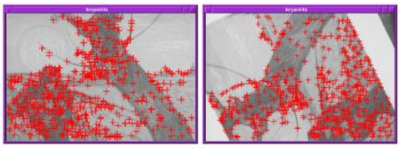
- *Those detectors have become a basic building block for many applications in Computer Vision.*

Keypoint Localization

- Goals:
  - Repeatable detection
  - Precise localization
  - Interesting content

  ⇒ *Look for two-dimensional signal changes*

Finding Corners



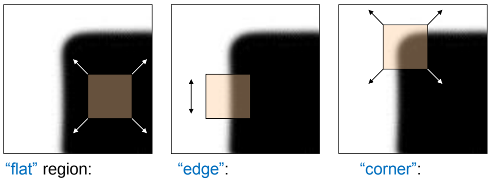
- Key property:
  - In the region around a corner, image gradient has two or more dominant directions
- Corners are *repeatable* and *distinctive*

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

Slide credit: Svetlana Lazebnik

Corners as Distinctive Interest Points

- Design criteria
  - We should easily recognize the point by looking through a small window (*locality*)
  - Shifting the window in *any direction* should give *a large change* in intensity (*good localization*)



“flat” region: no change in all directions	“edge”: no change along the edge direction	“corner”: significant change in all directions

Slide credit: Alexei Efros

Harris Detector Formulation

- Start from the second-moment matrix M :

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

Gradient with respect to x, times gradient with respect to y

Sum over image region – the area we are checking for corner

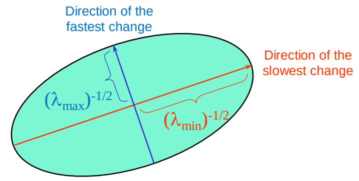
$$M = \begin{bmatrix} \sum I_xI_x & \sum I_xI_y \\ \sum I_xI_y & \sum I_yI_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y]$$

What Does This Matrix Reveal?

- Since M is symmetric, we have $M = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

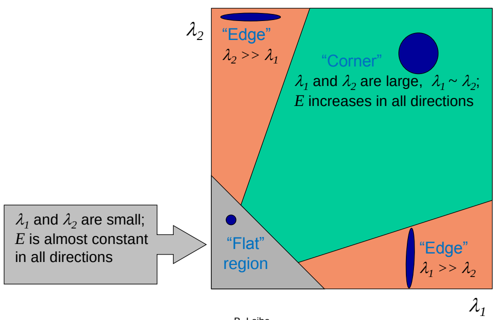
(Eigenvalue decomposition)

- We can visualize M as an ellipse with axis lengths determined by the eigenvalues and orientation determined by R



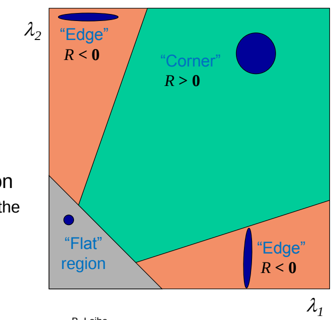
Interpreting the Eigenvalues

- Classification of image points using eigenvalues of M:



Corner Response Function

$$R = \det(M) - \alpha\, \mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$



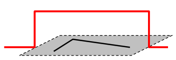
- Fast approximation
  - Avoid computing the eigenvalues
  - α: constant (0.04 to 0.06)

Window Function $w(x,y)$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

- Option 1: uniform window
  - Sum over square window

$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

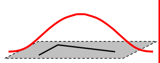


1 in window, 0 outside

  - Problem: not rotation invariant

- Option 2: Smooth with Gaussian
  - Gaussian already performs weighted sum

$$M = g(\sigma) * \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

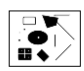


Gaussian

  - Result is rotation invariant

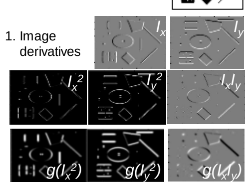
Summary: Harris Detector [Harris88]

- Compute second moment matrix (autocorrelation matrix)



$$M(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\ I_xI_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

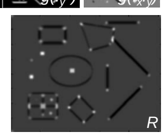
1. Image derivatives
2. Square of derivatives
3. Gaussian filter $g(\sigma_I)$



4. Cornerness function – two strong eigenvalues

$$R = \det[M(\sigma_I, \sigma_D)] - \alpha[\mathrm{trace}(M(\sigma_I, \sigma_D))]^2$$
$$= g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

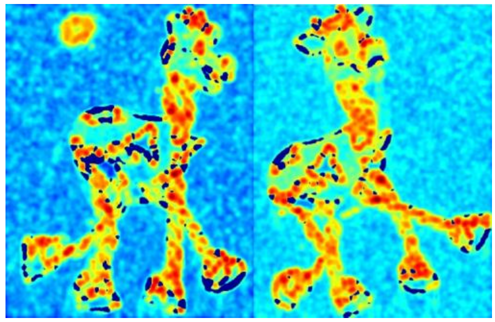
5. Perform non-maximum suppression



Harris Detector: Workflow

Computer Vision Summer'19

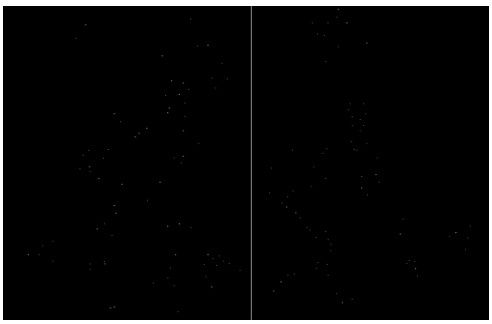
Harris Detector: Workflow



- Compute corner responses *R*

Computer Vision Summer'19

Harris Detector: Workflow



- Take only the local maxima of *R*, where *R* > threshold.

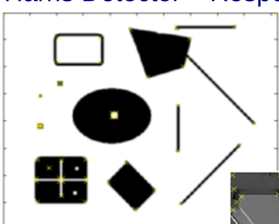
Computer Vision Summer'19

Harris Detector: Workflow

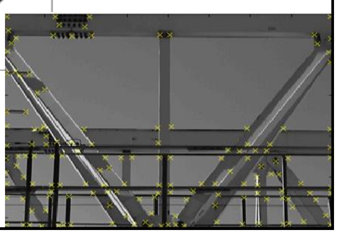
- Resulting Harris points

Computer Vision Summer'19

Harris Detector – Responses [Harris88]

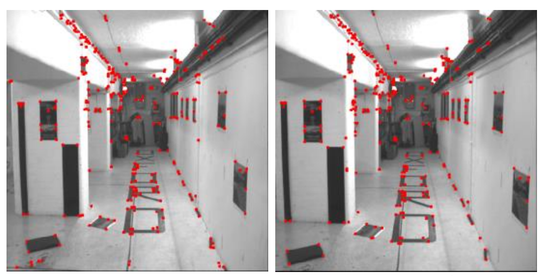


Effect: A very precise corner detector.



Computer Vision Summer'19

Harris Detector – Responses [Harris88]

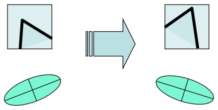


- Results are well suited for finding stereo correspondences

Computer Vision Summer'19

Harris Detector: Properties

- Rotation invariance?



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response R is invariant to image rotation

Computer Vision Summer'19

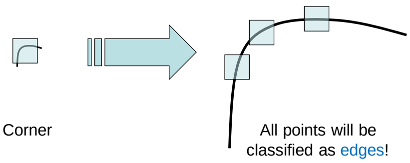
Slide credit: Kristen Grauman

B. Leibe

64

Harris Detector: Properties

- Rotation invariance
- Scale invariance?



Corner

All points will be classified as edges!

Not invariant to image scale!

Computer Vision Summer'19

Slide credit: Kristen Grauman

B. Leibe

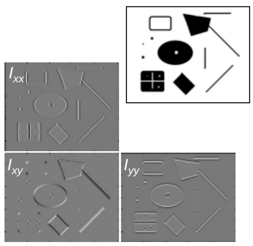
65

Hessian Detector [Beaudet78]

- Hessian determinant

$$Hessian(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

Note: these are 2nd derivatives!



Intuition: Search for strong derivatives in two orthogonal directions

Computer Vision Summer'19

Slide credit: Krystian Mikolajczyk

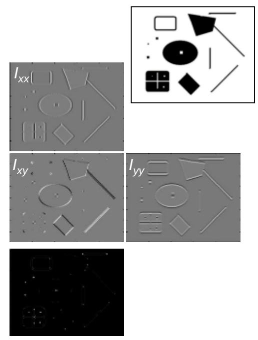
B. Leibe

66

Hessian Detector [Beaudet78]

- Hessian determinant

$$Hessian(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$



$$\det(Hessian(I)) = I_{xx} I_{yy} - I_{xy}^2$$

In Matlab:

$$I_{xx} .* I_{yy} - (I_{xy})\wedge 2$$

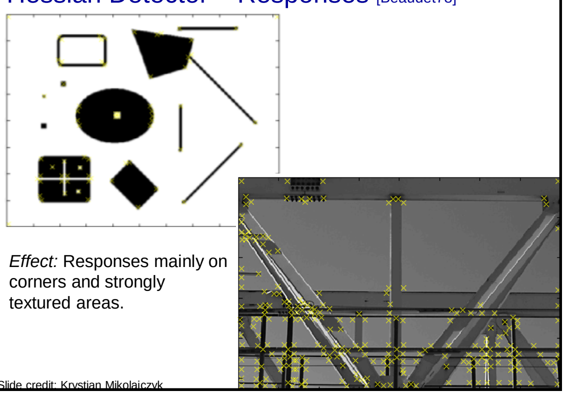
Computer Vision Summer'19

Slide credit: Krystian Mikolajczyk

B. Leibe

67

Hessian Detector – Responses [Beaudet78]



Effect: Responses mainly on corners and strongly textured areas.

Computer Vision Summer'19

Slide credit: Krystian Mikolajczyk

Topics of This Lecture

- Local Invariant Features
  - Motivation
  - Requirements, Invariances
- Keypoint Localization
  - Harris detector
  - Hessian detector
- Scale Invariant Region Selection
  - Automatic scale selection
  - Laplacian-of-Gaussian detector
  - Difference-of-Gaussian detector
  - Combinations
- Local Descriptors
  - Orientation normalization
  - SIFT

Computer Vision Summer'19

B. Leibe

70

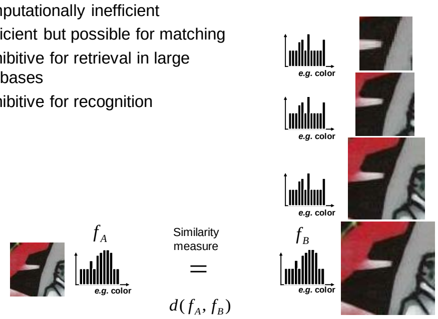
From Points to Regions…

- The Harris and Hessian operators define interest points.
  - Precise localization
  - High repeatability

- In order to compare those points, we need to compute a descriptor over a region.
  - How can we define such a region in a scale invariant manner?
- *I.e. how can we detect scale invariant interest regions?*

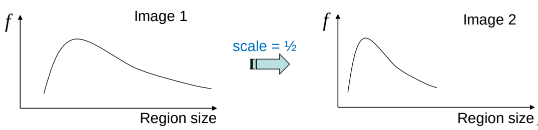
Naïve Approach: Exhaustive Search

- Comparing descriptors while varying the patch size
  - Computationally inefficient
  - Inefficient but possible for matching
  - Prohibitive for retrieval in large databases
  - Prohibitive for recognition



Automatic Scale Selection

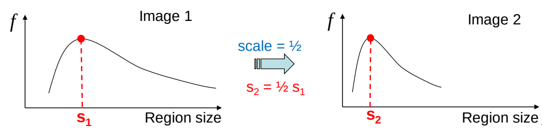
- Solution:
  - Design a signature function on the region that is "scale invariant" (*the same for corresponding regions, even if they are at different scales*)
  - For a point in one image, we can consider it as a function of region size (patch width)



Automatic Scale Selection

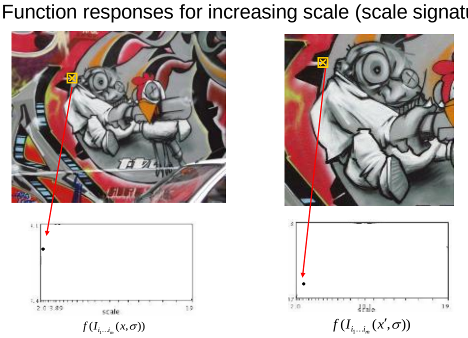
- Common approach:
  - Take a local maximum of this function.
  - Observation: region size for which the maximum is achieved should be *invariant* to image scale.

Important: this scale invariant region size is found in each image independently!



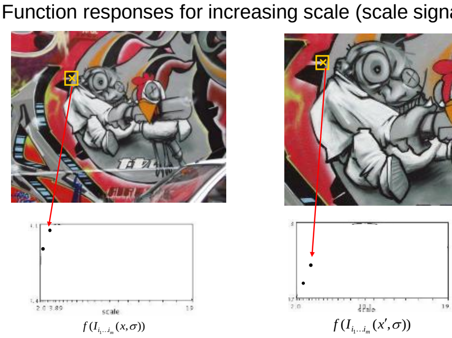
Automatic Scale Selection

- Function responses for increasing scale (scale signature)



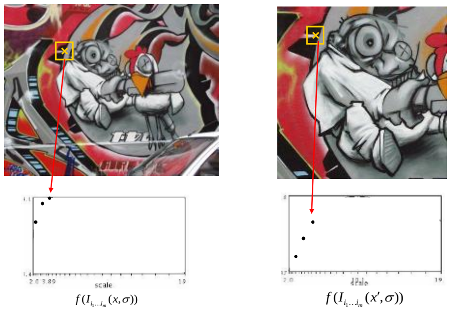
Automatic Scale Selection

- Function responses for increasing scale (scale signature)



Automatic Scale Selection

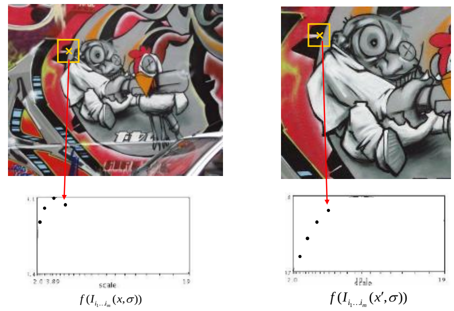
- Function responses for increasing scale (scale signature)



Slide credit: Krystian Mikolajczyk

Automatic Scale Selection

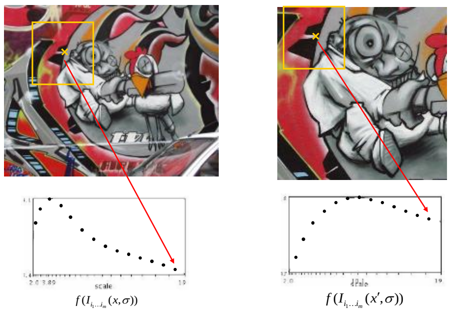
- Function responses for increasing scale (scale signature)



Slide credit: Krystian Mikolajczyk

Automatic Scale Selection

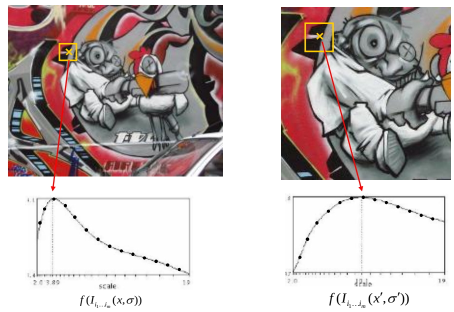
- Function responses for increasing scale (scale signature)



Slide credit: Krystian Mikolajczyk

Automatic Scale Selection

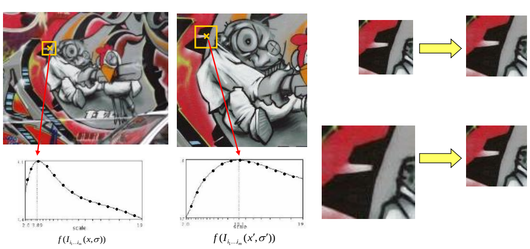
- Function responses for increasing scale (scale signature)



Slide credit: Krystian Mikolajczyk

Automatic Scale Selection

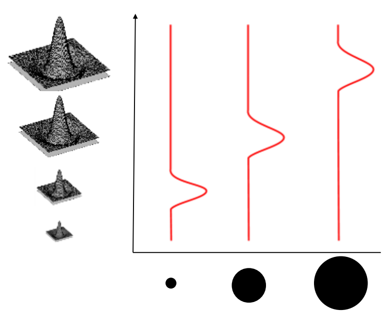
- Normalize: Rescale to fixed size



Slide credit: Tinne Tuytelaars

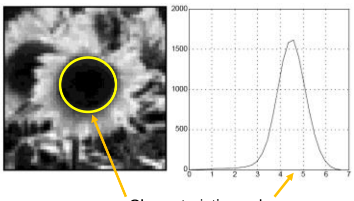
What Is A Useful Signature Function?

- Laplacian-of-Gaussian = "blob" detector



Characteristic Scale

- We define the *characteristic scale* as the scale that produces peak of Laplacian response

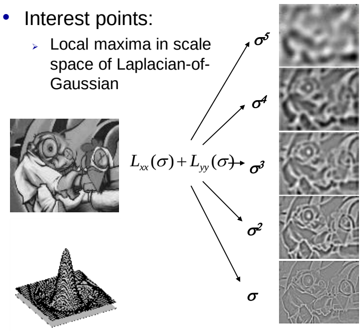


Characteristic scale

T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* 30 (2): pp 77--116.

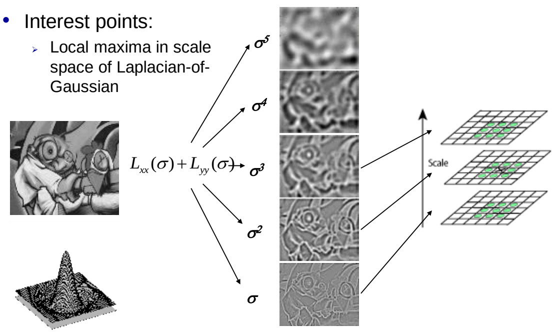
Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian



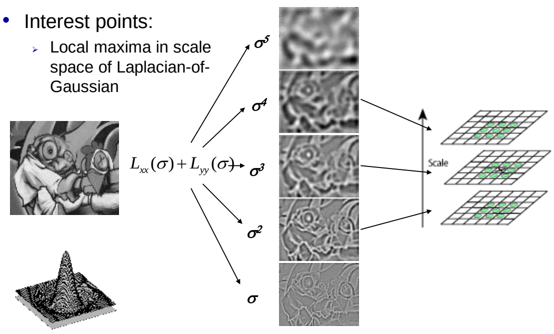
Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian



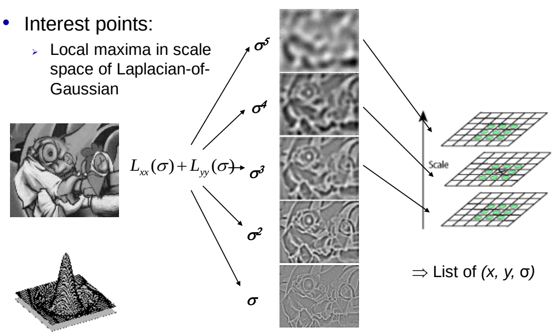
Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian



Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian



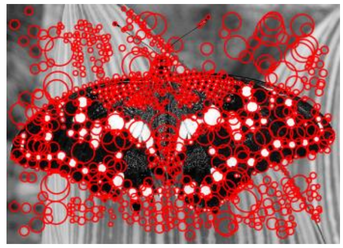
$\Rightarrow$ List of *(x, y, σ)*

LoG Detector: Workflow

LoG Detector: Workflow

sigma = 11.9912

LoG Detector: Workflow



Technical Detail

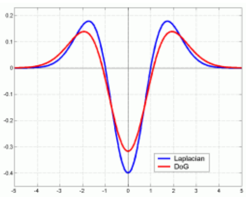
- We can efficiently approximate the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left(G_{xx}(x,y,\sigma) + G_{yy}(x,y,\sigma) \right)$$

(Laplacian)

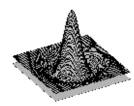
$$DoG = G(x,y,k\sigma) - G(x,y,\sigma)$$

(Difference of Gaussians)



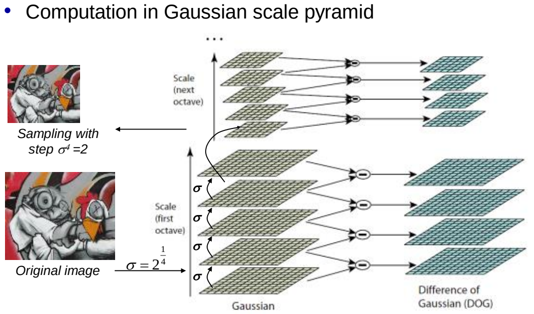
Difference-of-Gaussian (DoG)

- Difference of Gaussians as approximation of the LoG
  - This is used e.g. in Lowe's SIFT pipeline for feature detection.
- Advantages
  - No need to compute 2nd derivatives
  - Gaussians are computed anyway, e.g. in a Gaussian pyramid.



DoG – Efficient Computation

- Computation in Gaussian scale pyramid



Results: Lowe's DoG

Harris-Laplace [Mikolajczyk '01]

1. Initialization: Multiscale Harris corner detection
2. Scale selection based on Laplacian
   (same procedure with Hessian ⇒ Hessian-Laplace)

Harris points

Harris-Laplace points

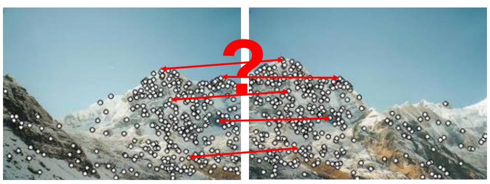
Summary: Scale Invariant Detection

- Given: Two images of the same scene with a large *scale difference* between them.
- Goal: Find *the same* interest points *independently* in each image.
- Solution: Search for *maxima* of suitable functions in *scale* and in *space* (over the image).
- Two strategies
  - Laplacian-of-Gaussian (LoG)
  - Difference-of-Gaussian (DoG) as a fast approximation
  - *These can be used either on their own, or in combinations with single-scale keypoint detectors (Harris, Hessian).*

Local Descriptors

- We know how to detect points
- Next question:

How to *describe* them for matching?



⇒ *Next lecture…*

You Can Try It At Home…

- For most local feature detectors, executables are available online:
- http://robots.ox.ac.uk/~vgg/research/affine
- http://www.cs.ubc.ca/~lowe/keypoints/
- http://www.vision.ee.ethz.ch/~surf
- http://homes.esat.kuleuven.be/~ncorneli/gpusurf/

http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries

References and Further Reading

- Read David Lowe's SIFT paper
  - D. Lowe,
    Distinctive image features from scale-invariant keypoints,
    IJCV 60(2), pp. 91-110, 2004
- Good survey paper on Int. Pt. detectors and descriptors
  - T. Tuytelaars, K. Mikolajczyk, *Local Invariant Feature Detectors: A Survey*, Foundations and Trends in Computer Graphics and Vision, Vol. 3, No. 3, pp 177-280, 2008.
- Try the example code, binaries, and Matlab wrappers
  - Good starting point: Oxford interest point page
    http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries