Computer Vision - Lecture 10

Local Features


Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de
leibe@vision.rwth-aachen.de
Course Outline

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Object Categorization I
  - Sliding Window based Object Detection
- Local Features & Matching
  - Local Features - Detection and Description
  - Recognition with Local Features
- Object Categorization II
  - Part based Approaches
- 3D Reconstruction
- Motion and Tracking
Recap: Sliding-Window Object Detection

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

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

Slide credit: Kristen Grauman
Recap: Gradient-based Representations

• Consider edges, contours, and (oriented) intensity gradients

• Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination
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

Slide credit: Kristen Grauman
Recap: Viola-Jones Face Detection

“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 \(\rightarrow\) scale features directly for same cost

Slide credit: Kristen Grauman

[Viola & Jones, CVPR 2001]
Recap: AdaBoost 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.

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.

Slide credit: Kristen Grauman

[Viola & Jones, CVPR 2001]
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/]
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**

**Randomized Forests**

Slide adapted from Kristen Grauman
Linear Classifiers

Let

\[ w = \begin{bmatrix} a \\ c \end{bmatrix}, \quad x = \begin{bmatrix} x \\ y \end{bmatrix} \]

\[ ax + cy + b = 0 \]

\[ w \cdot x + b = 0 \]
Linear Classifiers

- Find linear function to separate positive and negative examples

\[ x_i \text{ positive: } x_i \cdot w + b \geq 0 \]
\[ x_i \text{ negative: } x_i \cdot w + b < 0 \]

Which line is best?

Slide credit: Kristen Grauman
Support Vector Machines (SVMs)

- Discriminative classifier based on *optimal separating hyperplane* (i.e. line for 2D case)

- Maximize the *margin* between the positive and negative training examples

Slide credit: Kristen Grauman
Support Vector Machines

- Want line that maximizes the margin.

\[ \begin{align*}
    x_i \text{ positive } (y_i = 1) : & \quad x_i \cdot w + b \geq 1 \\
    x_i \text{ negative } (y_i = -1) : & \quad x_i \cdot w + b \leq -1
\end{align*} \]

For support vectors, \( x_i \cdot w + b = \pm 1 \)

**Quadratic optimization problem**

Minimize \( \frac{1}{2} w^T w \)

Subject to \( y_i (w \cdot x_i + b) \geq 1 \)

Finding the Maximum Margin Line

• Solution: \[ w = \sum_{i} \alpha_i y_i x_i \]
Finding the Maximum Margin Line

- **Solution:**
  \[ w = \sum_i \alpha_i y_i x_i \]
  \[ w \cdot x + b = \sum_i \alpha_i y_i x_i \cdot x + b \]

- **Classification function:**
  \[ f(x) = \text{sign} \left( w \cdot x + b \right) \]
  \[ = \text{sign} \left( \sum_i \alpha_i y_i x_i \cdot x + b \right) \]

  If \( f(x) < 0 \), classify as neg.,
  if \( f(x) > 0 \), classify as pos.

- Notice that this relies on an *inner product* between the test point \( x \) and the support vectors \( x_i \)

- (Solving the optimization problem also involves computing the inner products \( x_i \cdot x_j \) between all pairs of training points)

Non-Linear SVMs: Feature Spaces

• General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]

More on that in the Machine Learning lecture...

Slide from Andrew Moore’s tutorial: http://www.autonlab.org/tutorials/svm.html
Nonlinear SVMs

• *The kernel trick*: instead of explicitly computing the lifting transformation $\varphi(x)$, define a kernel function $K$ such that

$$K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$$

• This gives a nonlinear decision boundary in the original feature space:

$$\sum_i \alpha_i y_i K(x_i, x) + b$$

Some Often-Used Kernel Functions

- **Linear:**
  \[ K(x_i,x_j) = x_i^T x_j \]

- **Polynomial of power p:**
  \[ K(x_i,x_j) = (1 + x_i^T x_j)^p \]

- **Gaussian (radial-basis function):**
  \[ K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \]

Slide from Andrew Moore’s tutorial: [http://www.autonlab.org/tutorials/svm.html](http://www.autonlab.org/tutorials/svm.html)
Pedestrian detection with HoGs & SVMs

- **Navneet Dalal, Bill Triggs**, Histograms of Oriented Gradients for Human Detection, CVPR 2005
Summary: Sliding-Windows

• **Pros**
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes
  - Good detectors available (Viola & Jones, HOG, etc.)

• **Cons/Limitations**
  - High computational complexity
    - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
    - This puts tight constraints on the classifiers we can use.
    - If training binary detectors independently, this means cost increases linearly with number of classes.
  - With so many windows, false positive rate better be low
Limitations of Sliding Windows (continued)

- Not all objects are “box” shaped
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
Limitations (continued)

- If considering windows in isolation, context is lost

Figure credit: Derek Hoiem
Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions
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

B. Leibe
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
Harder Case

by Diva Sian

by scgbt

Slide credit: Steve Seitz
Harder Still?

NASA Mars Rover images

Slide credit: Steve Seitz
Answer Below (Look for tiny colored squares)

NASA Mars Rover images with SIFT feature matches (Figure by Noah Snavely)

Slide credit: Steve Seitz
Application: Image Stitching
Application: Image Stitching

- Procedure:
  - Detect feature points in both images
Application: Image Stitching

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

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

\[ d(f_A, f_B) < T \]
Common Requirements

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

No chance to match!

We need a repeatable detector!
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!
Invariance: Geometric Transformations

Slide credit: Steve Seitz
B. Leibe
Levels of Geometric Invariance

- Translation
- Euclidean
- Similarity
- Affine
- Projective
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 recent 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**

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: Alexej Efros*
Harris Detector Formulation

- Change of intensity for the shift \([u, v]\):

\[
E(u, v) = \sum_{x, y} w(x, y) \left[ I(x + u, y + v) - I(x, y) \right]^2
\]

Window function

Shifted intensity

Intensity

Window function \(w(x, y) = \) 1 in window, 0 outside

or

Gaussian

Slide credit: Rick Szeliski
Harris Detector Formulation

- This measure of change can be approximated by:

\[ E(u, v) \approx [u \ v] \ M [u \ v] \]

where \( M \) is a 2x2 matrix computed from image derivatives:

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

Gradient with respect to \( x \), times gradient with respect to \( y \)

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

\[
M = \begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix} = \sum \begin{bmatrix}
I_x \\
I_y
\end{bmatrix} [I_x \ I_y]
\]

Slide credit: Rick Szeliski
Harris Detector Formulation

where $M$ is a $2 \times 2$ matrix computed from image derivatives:

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

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

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

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x I_y]$$

Slide credit: Rick Szeliski
What Does This Matrix Reveal?

• First, let’s consider an axis-aligned corner:
What Does This Matrix Reveal?

- First, let’s consider an axis-aligned corner:

\[
M = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2
\end{bmatrix} = \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix}
\]

- This means:
  - Dominant gradient directions align with \( x \) or \( y \) axis
  - If either \( \lambda \) is close to 0, then this is not a corner, so look for locations where both are large.

- What if we have a corner that is not aligned with the image axes?
General Case

- 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 $R$

Slide credit: Kristen Grauman
adapted from Darya Frolova, Denis Simakov
Interpreting the Eigenvalues

- Classification of image points using eigenvalues of $M$:

\[
\begin{align*}
\lambda_1 & \text{ and } \lambda_2 \text{ are small; } E \text{ is almost constant in all directions} \\
\lambda_1 & \gg \lambda_2 \\
\lambda_1 & \text{ and } \lambda_2 \text{ are large, } \lambda_1 \sim \lambda_2; \\
E & \text{ increases in all directions} \\
\lambda_1 & \gg \lambda_2
\end{align*}
\]
Corner Response Function

\[ R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \]

- **Fast approximation**
  - Avoid computing the eigenvalues
  - \( \alpha \): constant (0.04 to 0.06)

Slide credit: Kristen Grauman
Window Function \( w(x, y) \)

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

- **Option 1: uniform window**
  - Sum over square window
  
    \[
    M = \sum_{x, y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}
    \]
  - Problem: not rotation invariant

- **Option 2: Smooth with Gaussian**
  - Gaussian already performs weighted sum
    
    \[
    M = g(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}
    \]
  - Result is rotation invariant
Summary: Harris Detector [Harris88]

- Compute second moment matrix (autocorrelation matrix)

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

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

4. Cornerness function - two strong eigenvalues

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

5. Perform non-maximum suppression

Slide credit: Krystian Mikolajczyk
Harris Detector: Workflow

Slide adapted from Darya Frolova, Denis Simakov, B. Leibe
Harris Detector: Workflow

- Compute corner responses $R$

Slide adapted from Darya Frolova, Denis Simakov B. Leibe
Harris Detector: Workflow

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

Slide adapted from Darya Frolova, Denis Simakov B. Leibe
Harris Detector: Workflow

- Resulting Harris points

Slide adapted from Darya Frolova, Denis Simakov B. Leibe
Harris Detector - Responses [Harris88]

**Effect:** A very precise corner detector.

Slide credit: Krystian Mikolajczyk
Harris Detector - Responses [Harris88]
Harris Detector - Responses [Harris88]

- Results are well suited for finding stereo correspondences

Slide credit: Kristen Grauman
Harris Detector: Properties

- Rotation invariance?

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

Corner response $R$ is invariant to image rotation
Harris Detector: Properties

- Rotation invariance
- Scale invariance?

Not invariant to image scale!

Corner

All points will be classified as edges!

Slide credit: Kristen Grauman
Hessian Detector [Beaudet78]

- Hessian determinant

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

Note: these are 2\textsuperscript{nd} derivatives!

**Intuition**: Search for strong derivatives in two orthogonal directions

Slide credit: Krystian Mikolajczyk
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:

\[
I_{xx} \ast I_{yy} - (I_{xy})^2
\]

Slide credit: Krystian Mikolajczyk
Hessian Detector - Responses [Beaudet78]

Effect: Responses mainly on corners and strongly textured areas.

Slide credit: Krystian Mikolajczyk
Hessian Detector - Responses [Beaudet78]
You Can Try It At Home...

- For most local feature detectors, executables are available online:
- [http://robots.ox.ac.uk/~vgg/research/affine](http://robots.ox.ac.uk/~vgg/research/affine)
- [http://www.vision.ee.ethz.ch/~surf](http://www.vision.ee.ethz.ch/~surf)
Affine Covariant Features

Affine Covariant Region Detectors

Input Image → Detector output → Image with displayed regions

Format:
1.0
m
u1 v1 a1 b1 c1
...
u m v m a m b m c m

Output example:
img1.harr

Parameters defining an affine region
u, v, a, b, c in a(x-u) + b(y-v) = c(x-u) + v(y-v) = 1
with (0,0) at image top left corner

Code
- provided by the authors, see publications for details and links to authors' web sites

Linux binaries
Harris-Affine & Hessian-Affine

Example of use
prompt>/h_affine.in -harraff -i img1.ppm -o img1.harrff -thres 1000

Displaying 1

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

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