Computer Vision II - Lecture 7

Tracking by Detection

15.05.2014

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de
Course Outline

• Single-Object Tracking
  - Background modeling
  - Template based tracking
  - Color based tracking
  - Contour based tracking
  - Tracking by online classification
  - Tracking-by-detection

• Bayesian Filtering

• Multi-Object Tracking

• Articulated Tracking

Image source: Helmut Grabner, Disney/Pixar
Today: Tracking by Detection

Object detections

Spacetime trajectories
Recap: Tracking as Online Classification

- Tracking as binary classification problem

object vs. background

Image source: Disney /Pixar

Slide credit: Helmut Grabner
Recap: Tracking as Online Classification

- Tracking as binary classification problem

- Handle object and background changes by online updating

Slide credit: Helmut Grabner

Image source: Disney /Pixar
Recap: AdaBoost - “Adaptive Boosting”

• Main idea
  - Iteratively select an ensemble of classifiers
  - Reweight misclassified training examples after each iteration to focus training on difficult cases.

• Components
  - \( h_m(x) \): “weak” or base classifier
    - Condition: <50% training error over any distribution
  - \( H(x) \): “strong” or final classifier

• AdaBoost:
  - Construct a strong classifier as a thresholded linear combination of the weighted weak classifiers:

\[
H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m h_m(x) \right)
\]
Recap: AdaBoost - Algorithm

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 $h_m(x)$ using the current weighting coefficients $W^{(m)}$ by minimizing the weighted error function
   \[
   J_m = \sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)
   \]

   b) Estimate the weighted error of this classifier on $X$:
   \[
   \epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(h_m(x) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}
   \]

   c) Calculate a weighting coefficient for $h_m(x)$:
   \[
   \alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}
   \]

   d) Update the weighting coefficients:
   \[
   w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(h_m(x_n) \neq t_n)\}
   \]
Recap: From Offline to Online Boosting

• Main issue
  - Computing the weight distribution for the samples.
  - We do not know a priori the difficulty of a sample! (Could already have seen the same sample before...)

• Idea of Online Boosting
  - Estimate the importance of a sample by propagating it through a set of weak classifiers.
  - This can be thought of as modeling the information gain w.r.t. the first \( n \) classifiers and code it by the importance weight \( \lambda \) for the \( n+1 \) classifier.
  - Proven [Oza]: Given the same training set, Online Boosting converges to the same weak classifiers as Offline Boosting in the limit of \( N \to \infty \) iterations.

Recap: From Offline to Online Boosting

**off-line**

**Given:**
- set of labeled training samples
  \( \mathcal{X} = \{\langle x_1, y_1 \rangle, \ldots, \langle x_L, y_L \rangle \mid y_i \pm 1 \} \)
- weight distribution over them
  \( D_0 = 1/L \)

for \( n = 1 \) to \( N \)
- train a weak classifier using samples and weight dist.
  \( h_{n}^{weak}(x) = \mathcal{L}(\mathcal{X}, D_{n-1}) \)
- calculate error \( e_n \)
- calculate weight \( \alpha_n = f(e_n) \)
- update weight dist. \( D_n \)
  next

\( h^{strong}(x) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_{n}^{weak}(x)) \)

**on-line**

**Given:**
- ONE labeled training sample
  \( \langle x, y \rangle \mid y \pm 1 \)
- strong classifier to update
- initial importance \( \lambda = 1 \)

for \( n = 1 \) to \( N \)
- update the weak classifier using samples and importance
  \( h_{n}^{weak}(x) = \mathcal{L}(h_{n}^{weak}, \langle x, y \rangle, \lambda) \)
- update error estimation \( \tilde{e}_n \)
- update weight \( \alpha_n = f(\tilde{e}_n) \)
- update importance weight \( \lambda \)
  next

\( h^{strong}(x) = \text{sign}(\sum_{n=1}^{N} \alpha_n \cdot h_{n}^{weak}(x)) \)

Slide credit: Helmut Grabner
Recap: Online Boosting for Feature Selection

- Introducing “Selector”
  - Selects one feature from its local feature pool
    
    \[
    \mathcal{H}^{weak} = \{h_1^{weak}, \ldots, h_M^{weak}\} \\
    \mathcal{F} = \{f_1, \ldots, f_M\} \\
    h^{sel}(x) = h_m^{weak}(x) \\
    m = \arg \min_i e_i
    \]

On-line boosting is performed on the Selectors and not on the weak classifiers directly.

H. Grabner and H. Bischof. 
On-line boosting and vision. 
Recap: Direct Feature Selection

- **Shared feature pool for all selectors to save computation**

Slide credit: Helmut Grabner
Recap: Tracking by Online Classification

- From time $t$ to $t+1$
- Evaluate classifier on sub-patches
- Search region
- Create confidence map
- Analyze map and set new object position
- Update classifier (tracker)
- Actual object position
- Image source: Disney /Pixar

Slide credit: Helmut Grabner
Recap: Self-Learning and Drift

- **Drift**
  - Major problem in all adaptive or self-learning trackers.
  - Difficulty: distinguish “allowed” appearance changes due to lighting or viewpoint variation from “unwanted” appearance change due to drifting.
  - Cannot be decided based on the tracker confidence!

- **Several approaches to address this**
  - Comparison with initialization
  - Semi-supervised learning (additional data)
  - Additional information sources
Topics of This Lecture

• Tracking by Detection
  ➢ Motivation
  ➢ Recap: Object detection

• SVM based Detectors
  ➢ Recap: HOG
  ➢ DPM

• AdaBoost based Detectors
  ➢ Recap: Viola-Jones
  ➢ Integral Channel features
  ➢ VeryFast/Roerei

• Random Forest based Detectors
  ➢ Recap: ISM
  ➢ Hough Forests
Detection-Based Tracking

Main ideas

- Apply a generic object detector to find objects of a certain class
- Based on the detections, extract object appearance models
  - Even possible to derive figure-ground segmentations from detection results
- Link detections into trajectories
Tracking-by-Detection in 3D

Object detections

3D Camera path estimation

Spacetime trajectories

Main Issue: Data Association
(We’ll come to that...)

Simple f/g model:
E.g., elliptical region in detection box

[Leibe, Cornelis, Schindler, Van Gool, PAMI’08]
Spacetime Trajectory Analysis

Pedestrian detection

Car detections

Own vehicle

[Leibe, Cornelis, Cornelis, Van Gool, CVPR’07]
Elements of Tracking

- Detection
  - Where are candidate objects?

- Data association
  - Which detection corresponds to which object?

- Prediction
  - Where will the tracked object be in the next time step?

Today’s topic
Recap: Sliding-Window Object Detection

- Basic component: a binary classifier
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
What is a Sliding Window Approach?

- Search over space and scale

- Detection as subwindow classification problem

- “In the absence of a more intelligent strategy, any global image classification approach can be converted into a localization approach by using a sliding-window search.”
Recap: Non-Maximum Suppression

After multi-scale dense scan

Goal

Fusion of multiple detections

Clip detection score

Map each detection to 3D \([x,y,\text{scale}]\) space

Apply robust mode detection, \textit{e.g.} mean shift

Non-maximum suppression

Image source: Navneet Dalal, PhD Thesis
Recap: Sliding-Window Object Detection

Fleshing out this pipeline a bit more, we need to:

1. Obtain training data
2. Define features
3. Define classifier
Object Detector Design

• In practice, the classifier often determines the design.
  - Types of features
  - Speedup strategies

• Today, we’ll look at 3 state-of-the-art detector designs
  - Based on SVMs
  - Based on Boosting
  - Based on Random Forests
Topics of This Lecture

• Tracking by Detection
  ➢ Motivation
  ➢ Recap: Object detection

• SVM based Detectors
  ➢ Recap: HOG
  ➢ DPM

• AdaBoost based Detectors
  ➢ Recap: Viola-Jones
  ➢ Integral Channel features
  ➢ VeryFast/Roerei

• Random Forest based Detectors
  ➢ Recap: ISM
  ➢ Hough Forests
Recap: Histograms of Oriented Gradients (HOG)

- Holistic object representation
  - Localized gradient orientations
    \[
    \left[ \ldots, \ldots, \ldots, \ldots \right]
    \]

Slide adapted from Navneet Dalal
Recap: Support Vector Machine (SVM)

- Basic idea
  - The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
  - Up to now: consider linear classifiers
    \[ w^T x + b = 0 \]

- Formulation as a convex optimization problem
  - Find the hyperplane satisfying
    \[ \arg\min_{w,b} \frac{1}{2} \|w\|^2 \]
    under the constraints
    \[ t_n (w^T x_n + b) \geq 1 \quad \forall n \]
    based on training data points \( x_n \) and target values \( t_n \in \{-1, 1\} \).
Recap: Pedestrian Detection with HOG

- Train a pedestrian template using a linear SVM
- At test time, convolve feature map with template

\[ y(x) = w^T x + b \]

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

Slide credit: Svetlana Lazebnik
Pedestrian detection with HoGs & SVMs

- N. Dalal, B. Triggs, Histograms of Oriented Gradients for Human Detection, CVPR’05

Slide credit: Kristen Grauman
Extensions and Improvements(?)

- Choice of evaluation criterion is critical!
  - Traditional evaluations on per-window classification.
  - [Dollar et al.,’09]: *None* of the methods proposed from 2004-2009 brought an improvement for the actual detection task!

(a) INRIA per-window results.
(b) INRIA per-image results.
Some Extensions that Did Survive…

- **HOG + LBP**  
  - Compute LBP histograms over cells, as in HOG  
  - Features seem to be complementary to some degree  
  - Ojala & Pietikäinen 1999, Wang et al. ‘09

- **HOG + Depth + Flow**  
  - For applications in intelligent vehicles where those are available  
  - Factor 40 reduction in false positives possible  
  - Wojek et al. 2010, Gavrila 2012

- **HIK-SVM**  
  - Apply non-linear SVM kernels at reduced cost  
  - Maji et al. 2008

- **Explicit Feature Maps**  
  - Same as above, but on steroids  
  - Vedaldi & Zisserman 2010, ‘12
Incorporating Ground Plane Constraints

- Efficient integration into detector design (**groundHOG**)
  - Idea: only evaluate geometrically valid detection windows
  - Derivation: Region of interest lies between two parabolas...
  - ...that can in most cases be approximated by straight lines.
  ⇒ Only touch pixels inside the ROI for all computations.
  ⇒ Factor 2-4 speed improvement on top of all other optimizations

[P. Sudowe, B. Leibe, ICVS’11]
Real-Time Pedestrian Detection

- Efficient CUDA HOG implementation (equivalent to original HOG code)
- Code made publicly available as open source under GPL
- Run-time comparison:

<table>
<thead>
<tr>
<th>run-time</th>
<th>1280 × 960</th>
<th></th>
<th>640 × 480</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cuda</td>
<td>ground</td>
<td>cuda</td>
<td>ground</td>
</tr>
<tr>
<td>Laptop GTX 285M</td>
<td>1.6 fps</td>
<td>9.6 fps</td>
<td>7.2 fps</td>
<td>26 fps</td>
</tr>
<tr>
<td>Desktop GTX 280</td>
<td>5.5 fps</td>
<td>17.2 fps</td>
<td>22.7 fps</td>
<td>56 fps</td>
</tr>
<tr>
<td>Desktop GTX 580</td>
<td>9.8 fps</td>
<td>27.8 fps</td>
<td>41.6 fps</td>
<td>83 fps</td>
</tr>
</tbody>
</table>

⇒ Detection at video frame rate possible even on laptops with mobile GPUs!
You Can Try It At Home...

- groundHOG GPU detector code publicly available
  - Highly optimized for speed
  - Can be used with or without ground plane constraints
  - Supports general ROI processing
  - Supports multi-class detection with feature sharing
  - Published under GPL license (other licensing negotiable)

- [http://www.vision.rwth-aachen.de/projects/groundhog](http://www.vision.rwth-aachen.de/projects/groundhog)

**P. Sudowe, B. Leibe,** *Efficient Use of Geometric Constraints for Sliding Window Object Detection in Video*, ICVS 2011
Topics of This Lecture

• Tracking by Detection
  ➢ Motivation
  ➢ Recap: Object detection

• **SVM based Detectors**
  ➢ Recap: HOG
  ➢ DPM

• **AdaBoost based Detectors**
  ➢ Recap: Viola-Jones
  ➢ Integral Channel features
  ➢ VeryFast/Roerei

• **Random Forest based Detectors**
  ➢ Recap: ISM
  ➢ Hough Forests
Recap: Part-Based Models

- Pictorial Structures model
  - [Fischler & Elschlager 1973]

- Model has two components
  - Parts (2D image fragments)
  - Structure (configuration of parts)

- Use in Deformable Part-based Model (DPM)
  - Parts \( \equiv \) 5-7 semantically meaningful parts
  - Probabilistic model enabling efficient inference
Starting Point: HOG Sliding-Window Detector

- Array of weights for features in window of HOG pyramid
- Score is dot product of filter and vector

Filter $F$

Score of $F$ at position $p$ is $F \cdot \phi(p, H)$

$\phi(p, H) = \text{concatenation of HOG features from window specified by } p.$
Deformable Part-based Models

- Mixture of deformable part models (Pictorial Structures)
- Each component has global template + deformable parts
- Fully trained from bounding boxes alone

[Slide credit: Pedro Felzenszwalb]
2-Component Bicycle Model

Root filters
coarse resolution

Part filters
finer resolution

Deformation models

Slide credit: Pedro Felzenszwalb

B. Leibe

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Object Hypothesis

Multiscale model captures features at two resolutions

Score of object hypothesis is sum of filter scores minus deformation costs

Score of filter: dot product of filter with HOG features underneath it

Image pyramid

HOG feature pyramid

Slide credit: Pedro Felzenszwalb

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Score of a Hypothesis

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot (dx_i^2, dy_i^2)
\]

- "data term"
- "spatial prior"
- filters
- displacements
- deformation parameters

\[
\text{score}(z) = \beta \cdot \Psi(H, z)
\]

- concatenation filters and deformation parameters
- concatenation of HOG features and part displacement features

Slide credit: Pedro Felzenszwalb

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Recognition Model

\[ f_w(x) = w \cdot \Phi(x) \]

\[ f_w(x) = \max_z w \cdot \Phi(x, z) \]

- **Difference to standard HOG model**
  - Hidden variable \( z \): vector of part offsets
  - \( \Phi(x, z) \): vector of HOG features (from root filter & appropriate part sub-windows) and part offsets
  \[ \Rightarrow \text{Need to optimize over all possible part positions} \]

Slide adapted from Pedro Felzenszwalb

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
feature map

model

feature map at twice the resolution

response of part filters

transformed responses

color encoding of filter response values

combined score of root locations

B. Leibe

Slide credit: Pedro Felzenszwalb
Results: Persons

- Results (after non-maximum suppression)
  - ~1s to search all scales

Slide credit: Pedro Felzenszwalb

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Results: Bicycles

Slide adapted from Trevor Darrell

B. Leibe

[Felzenszwalb, McAllister, Ramanan, CVPR’08]
Extensions and Detailed Improvements

• More efficient features
  ➢ Very simplified version of HOG

• Latent part (re-)learning
  ➢ Perform several rounds of training, adapting the annotation bboxes

• Multi-aspect detection
  ➢ Mixture model of different aspects to capture different viewpoints of objects

• Bounding box prediction
  ➢ Infer final detection bounding box from detected part locations

• Multi-resolution models

• Cascaded evaluation

Felzenszwalb, McAllister, Ramanan, PAMI’10
You Can Try It At Home...

- Deformable part-based models have been very successful at several recent evaluations.
  \[\Rightarrow\] One of the state-of-the-art approaches in object detection

- Source code and models trained on PASCAL 2006, 2007, and 2008 data are available here:
  \[
  \text{http://www.cs.uchicago.edu/~pff/latent}
  \]
Topics of This Lecture

• Tracking by Detection
  - Motivation
  - Recap: Object detection

• SVM-based Detectors
  - Recap: HOG
  - DPM

• AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei

• Random Forest based Detectors
  - Recap: ISM
  - Hough Forests
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/]

Slide credit: Kristen Grauman
Recap: Haar Wavelets

“Rectangular” filters

Feature output is difference between adjacent regions

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

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

Avoid scaling images ⇒ Scale features directly for same cost

\[
D = 1 + 4 - (2 + 3) = A + (A + B + C + D) - (A + C + A + B) = D
\]

Integral image

Slide credit: Kristen Grauman

[Viola & Jones, CVPR 2001]
AdaBoost for Efficient Feature Selection

- Image features = weak classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples

Recap: Cascading Classifiers for Detection

- Even if the filters are fast to compute, each new image has a lot of possible windows to search...

- **Idea: Classifier cascade**
  - Observation: most image windows are negative and look very different from the searched object class.
  - 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’01; Rowley et al., PAMI’98; Viola & Jones, CVPR’01]
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/](http://sourceforge.net/projects/opencvlibrary/)

- Matlab wrappers for OpenCV code available, e.g. here

P. Viola, M. Jones, [Robust Real-Time Face Detection](http://www.cs.cmu.edu/~abanerjee/Projects/face_detection/robust_realtime_face_detection.pdf), IJCV, Vol. 57(2), 2004

Slide credit: Kristen Grauman
Topics of This Lecture

• Tracking by Detection
  - Motivation
  - Recap: Object detection

• SVM-based Detectors
  - Recap: HOG
  - DPM

• AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei

• Random Forest based Detectors
  - Recap: ISM
  - Hough Forests
Integral Channel Features

- Generalization of Haar Wavelet idea from Viola-Jones
  - Instead of only considering intensities, also take into account other feature channels (gradient orientations, color, texture).
  - Still efficiently represented as integral images.

Integral Channel Features

- Generalize also block computation
  - 1\textsuperscript{st} order features:
    - Sum of pixels in rectangular region.
  - 2\textsuperscript{nd}-order features:
    - Haar-like difference of sum-over-blocks
  - Generalized Haar:
    - More complex combinations of weighted rectangles
  - Histograms
    - Computed by evaluating local sums on quantized images.
Results: Integral Channel Features

  - fastHOG
  - ~10 Hz on GPU
  - [Prisacariu 2009]
- Felzenszwalb (2008)
  - DPM
  - [Felzenszwalb 2008]
- Dollar (2009+2010)
  - ChnFtrs/FPDW
  - ~5 Hz on CPU
  - [Dollar 2009+2010]

Slide credit: Rodrigo Benenson
Topics of This Lecture

- Tracking by Detection
  - Motivation
  - Recap: Object detection
- SVM-based Detectors
  - Recap: HOG
  - DPM
- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei
- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests
INRIA dataset

VeryFast 50 Hz

Slide credit: Rodrigo Benenson
Issues for Efficient Detection

• One template cannot detect at multiple scales...
Issues for Efficient Detection

- Typically, features are computed many times

~50 scales

Slide credit: Rodrigo Benenson

B. Leibe
Issues for Efficient Detection

- Typically, features are computed many times
**VeryFast Detector**

- Idea 1: Invert the relation


Slide credit: Rodrigo Benenson
Practical Considerations

- Training and running 1 model/scale is too expensive
VeryFast Detector

• Idea 2: Reduce training time by feature interpolation

5 models, 1 image scale

≈

50 models, 1 image scale

• Shown to be possible for Integral Channel features
VeryFast Detector

- Effect: Transfer test time computation to training time

⇒ Result: 3x reduction in feature computation
VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ score = w_1 \cdot h_1 + \]

Slide credit: Rodrigo Benenson
VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ \text{score} = w_1 \cdot h_1 + w_2 \cdot h_2 + \]

Slide credit: Rodrigo Benenson
VeryFast: Classifier Construction

- Ensemble of short trees, learned by AdaBoost

\[ \text{score} = w_1 \cdot h_1 + w_2 \cdot h_2 + \ldots + w_N \cdot h_N \]

Slide credit: Rodrigo Benenson
Learned Models

Integral Channel features

Models

Slide adapted from Rodrigo Benenson
Results

- Detection without resizing provides quality

Slide adapted from Rodrigo Benenson
Multi-Scale Models > Single-Scale Model

Slide adapted from Rodrigo Benenson
Comparison to State-of-the-Art

• **Extension: Roerei detector**
  - Detailed evaluation of design space
  - Non-regular pooling regions found to work best.

Slide adapted from Rodrigo Benenson
Roerei Results

Applications: Mobile Robot Navigation

link to the video
Topics of This Lecture

- Tracking by Detection
  - Motivation
  - Recap: Object detection

- SVM-based Detectors
  - Recap: HOG
  - DPM

- AdaBoost based Detectors
  - Recap: Viola-Jones
  - Integral Channel features
  - VeryFast/Roerei

- Random Forest based Detectors
  - Recap: ISM
  - Hough Forests
Recap: Implicit Shape Model (ISM) Idea

- Visual vocabulary is used to index votes for object position [a visual word = “part”].

Recap: Implicit Shape Model (ISM) Idea

- Objects are detected as consistent configurations of the observed parts (visual words).

Test image

Recap: ISM - Representation

- Learn appearance codebook
  - Extract local features at interest points
  - Feature clustering \(\Rightarrow\) codebook

- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object

Training images
(+reference segmentation)

Appearance codebook

Spatial occurrence distributions
+ local figure-ground labels
Recap: ISM - Recognition

Interest Points → Matched Codebook Entries → Probabilistic Voting

Image Feature → Interpretation (Codebook match) → Object Position

\[ f \rightarrow C_i \rightarrow o,x \]

\[ p(C_i|f) \rightarrow p(o_n, x|C_i, \ell) \]

Probabilistic vote weighting

[Leibe, Leonardis, Schiele, SLCV’04; IJCV’08]
Recap: ISM - Recognition

- Interest Points
- Matched Codebook Entries
- Probabilistic Voting

3D Voting Space (continuous)

Backprojected Hypotheses

Backprojection of Maxima

[Leibe, Leonardis, Schiele, SLCV’04; IJCV’08]
Recap: ISM - Top-Down Segmentation

Interest Points

Matched Codebook Entries

Probabilistic Voting

Segmentation

$p(figure)$ Probabilities

Backprojected Hypotheses

Backprojection of Maxima

[Leibe, Leonardis, Schiele, SLCV’04; IJCV’08]
Class-Specific Top-Down Segmentation

• During initial Hough Voting
  - When we first observe a feature, we do not know its context.
  - Different figure-ground labels may be consistent with the appearance.
  ⇒ Strategy: we cast votes for many locations...

• After voting
  - Voting groups features that are consistent with the same object.
  - We can now consider each feature conditioned on the selected object location hypothesis.
  - This allows us to backproject a local figure-ground label from selected votes.

B. Leibe
Top-Down Segmentation

- Interpretation of $p(\text{figure})$ map
  - per-pixel confidence in object hypothesis
  - Useful for hypothesis verification

Original image

$p(\text{figure})$

$p(\text{ground})$

Segmentation

$p(\text{figure})$

$p(\text{ground})$

[Leibe, Leonardis, Schiele, SLCV’04; IJCV’08]
Recap: ISM - Example Results

[Leibe, Leonardis, Schiele, SLCV’04; IJCV’08]
Topics of This Lecture

• Tracking by Detection
  ➢ Motivation
  ➢ Recap: Object detection

• SVM-based Detectors
  ➢ Recap: HOG
  ➢ DPM

• AdaBoost based Detectors
  ➢ Recap: Viola-Jones
  ➢ Integral Channel features
  ➢ VeryFast/Roerei

• Random Forest based Detectors
  ➢ Recap: ISM
  ➢ Hough Forests
Hough Forest Object Detector

- Combine idea of ISM-style Hough voting with dense feature sampling and discriminative training.
  - Randomized forest classifier densely processes image patches
  - Leaf nodes correspond to visual words
  - Cast votes for possible object hypotheses
- Good empirical performance, fast to evaluate

[Source: Gall, CVPR’09]
Fast Dense Matching with Random Forests

• Ideas
  - Solve feature extraction and codebook matching at the same time
  - Discriminative training of codebook features

• Extremely simple features
  - 2-pixel comparisons in different feature channels
  - Evaluation sub-linear in patch size

• Tree construction
  - Each leaf node contains occurrence distribution for Hough Voting
  - Training goal: Minimize class entropy while keeping distributions compact
Multi-View Extension

- **Random Forests are implicitly multi-class capable**
  - Create multi-class tree with per-class occurrence distributions
  - Use one Hough space per class or viewpoint
  - Necessary: multi-class non-maximum suppression

[Razavi, Alvar, Gall, van Gool, CVPR’11; Rematas, Leibe, CORP’11]
Top-Down Segmentation with Hough Forests

- Extend HFs with top-down segmentation mechanism
- Better results than for ISM due to dense sampling

[Rematas, Leibe, CORP’11]
HF-ISM: Qualitative Results

• Observations
  - Improved detection performance compared to original HF (competitive with HOG + HIKSVM on pedestrians).
  - Better segmentations than original ISM due to dense sampling.

(no ground plane constraints used)
You Can Try All of This At Home...

- Detector code is publicly available
  - HOG: Dalal’s original implementation: [http://www.navneetdalal.com/software/](http://www.navneetdalal.com/software/)
    - Our CUDA-optimized groundHOG code (>80 fps on GTX 580): [http://www.mmp.rwth-aachen.de/projects/groundhog](http://www.mmp.rwth-aachen.de/projects/groundhog)
  - DPM: Felzenswalb’s original implementation: [http://www.cs.uchicago.edu/~pff/latent](http://www.cs.uchicago.edu/~pff/latent)
  - ISM: My original implementation: [http://www.vision.rwth-aachen.de/software/ism](http://www.vision.rwth-aachen.de/software/ism)
  - HF: Gall’s original implementation: [http://www.vision.ee.ethz.ch/~gallju/index.html#software](http://www.vision.ee.ethz.ch/~gallju/index.html#software)
  - VeryFast: Benenson’s original implementation: [https://bitbucket.org/rodrigob/doppia/](https://bitbucket.org/rodrigob/doppia/)